

版权注意事项：1、书籍版权归著者和出版社所有；
2、本PDF仅用于个人获取知识，进行私底下知识交流；
3、PDF获得者不得在互联网以任何目的进行传播；
如有需要，请尽量购买正版实体书！支持书籍作者！！

PEARSON

Broadview®
www.broadview.com.cn

自己动手 设计数据库

Database Design

for Mere Mortals (3rd Edition)

A Hands-On Guide to Relational Database Design

[美] Michael J. Hernandez 著 • 盛杨燕 译



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

作者简介 /

迈克尔·J·埃尔南德斯 (Michael J. Hernandez) 是一个拥有二十余年丰富经验的关系数据库开发人员，同时也是微软开发工具 (Microsoft Visual Studio) 团队的项目经理和产品经理。他一直是如AppDev培训有限公司、焦点集团 (Focal Point, Inc.) 和深度培训 (Deep Training) 等公司的知名讲师，也是全美、欧洲和南美的技术会议上的一流演讲者。他还与约翰·L·维斯卡斯 (John L. Viescas) 一起合著了《SQL查询凡人入门》(第2版) (Addison-Wesley, 2008)。

自己动手 设计数据库

Database Design
for Mere Mortals (3rd Edition)
A Hands-On Guide to Relational Database Design

[美] Michael J. Hernandez 著 盛杨燕 译

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书主要讲述数据库的设计,讨论了如何建立表结构、确定主键、设置字段说明、建立表关系、确立业务规则、建立视图和各层次的数据完整性,以及如何避免不好的设计等问题。本书提供的是数据库设计的一种概念性思路,因此与市面上众多的同类书籍相比,本书有两个比较鲜明的特点。第一,作者采用简单易懂的语言,尽量清晰、全面地描述关系数据库设计的整个过程,没有过多专业的术语和复杂的数据库设计方法学,因此本书既适合专业人士参考之用,也适合给初学者、数据库设计爱好者充当从入门到进阶的重要读物。第二,作者高度重视数据库的逻辑设计,严格区分逻辑设计和实现阶段,以确保高效、成功地设计良好的数据库。

本书适合数据库初学者、有经验的数据库开发人员,以及所有对数据库设计感兴趣的读者阅读参考。

Authorized translation from the English language edition, entitled DATABASE FOR MERE MORTALS: A HANDS-ON GUIDE TO RELATIONAL DATABASE DESIGN, 3E, by HERNANDEZ, MICHAEL J., published by Pearson Education, Inc., publishing as Addison-Wesley Professional, Copyright © 2013 Michael J. Hernandez.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD., and PUBLISHING HOUSE OF ELECTRONICS INDUSTRY Copyright © 2015.

本书简体中文版专有出版权由 Pearson Education 培生教育出版亚洲有限公司授予电子工业出版社。未经出版者预先书面许可,不得以任何方式复制或抄袭本书的任何部分。

本书简体中文版贴有 Pearson Education 培生教育集团激光防伪标签,无标签者不得销售。

版权贸易合同登记号 图字:01-2014-6160

图书在版编目(CIP)数据

自己动手设计数据库 / (美) 埃尔南德斯 (Hernandez, M.J.) 著; 盛杨燕译. —北京: 电子工业出版社, 2015.9
书名原文: Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design (3rd Edition)
ISBN 978-7-121-26532-7

I. ①自… II. ①埃… ②盛… III. ①关系数据库—程序设计 IV. ①TP311.132.3

中国版本图书馆 CIP 数据核字 (2015) 第 149011 号

策划编辑: 张春雨 符隆美

责任编辑: 白 涛

印 刷: 北京中新伟业印刷有限公司

装 订: 三河市良远印务有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本: 787×980 1/16 印张: 31 字数: 585 千字

版 次: 2015 年 9 月第 1 版

印 次: 2016 年 6 月第 2 次印刷

定 价: 99.00 元

凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888, 88258888。

质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式: 010-51260888-819 faq@phei.com.cn。

乳升——乳甜贱学要公升代印姓

乳升——乳甜贱学要公升代印姓

献给我的妻子，感谢她始终如一地信任我。

献给一路帮助过我的人——我的老师、导师、朋友和同事。

献给所有曾经尝试过设计关系数据库但失败了的人。

我们为什么要学数据库——代序



我从小就很喜欢读书，但不明白书有好坏之分，恰好父亲淘了两大箱子文化宫处理的旧书，于是我就背着父亲一本一本从箱子里面偷书出来看，看完了之后再胡乱塞回去。刚开始偷书看的时候也就三四岁，应该也看过很多“少儿不宜”的书，我记得小学一年级前就开始看梁羽生的《弹指惊雷》，也不记得那个时候看懂了多少。看完之后的书塞回箱子顶部，也从来没有整理过，所以到后来伸手进去摸（箱子搬不出来），顶上都是些看过的书，要找到一本还没有看过的书，是越来越不容易。

后来离开家出去读书，不管是在中国科学技术大学还是在瑞士弗里堡大学，我都有让同学艳羡的“藏书”。那时候一箱一箱搬运到弗里堡大学的中文书，应该还在造福来自中国的学弟学妹。和父亲越来越乱的大书箱一样，因为没有条件，所以喜欢不喜欢的书，课外课内的书，看过没看过的书，都杂乱地堆在一起。只是因为是自己选过、买过、品过的东西，总是能够很快找到所需要的书，若是换了另外一个人找我借一本教材，恐怕是很难从“垃圾山”一般的书堆中自己找到想要的教材。

2013年，我第一次有了一个自己独立的办公室。办公室面积不大，但是在我的坚持下，配备了一个很大的书柜。书柜有4层书架，可以放一百二十本书，这些书放在书架上，直接就可以取阅，来办公室的客人也能够看到书的名字。书架两侧还各有一扇木头的拉门，里面是柜子和架子，挤挤各自可以放两百来本书，但是从外面看不到，取阅也不太方便。我选了三类书放在明面的书架上，一是自己特别喜欢，也特别能表达自己兴趣的书；二是短

期内可能就会阅读的书，一般都是新近购买的书籍；三是经常使用的工具书。其他用得相对较少的书，或者刚刚看过的一些书，就只能退让一步，躲到拉门后面了。

对于一个爱书的人，怎么样排列自己的书，成了一个幸福的问题。书排列得好，既方便一下子找到想要的书，也方便购买新书后能够一下子插入到合适的位置，不至于书越多越乱。可以用来排列图书的办法很多，例如按照出版社的名字、按照作者的名字、按照图书的题目等，甚至可以按照图书书脊的颜色、图书的重量、页码、定价、字数等。我有一次到一个朋友家，看到他家卧室里面一个很小的书柜，也就五十本书的规模吧，竟然是按照颜色排列的。我好奇地问他为什么如此， he 说是小女儿排的，而且非要如此，一旦有些变动，就又哭又闹的。

图书的重量、字数等，其实是很好的标识，因为重量或者字数完全一样的书很少，基本上可以唯一确定一本书。但是用这个来排列图书并不合适，一是要随时确定图书的重量和字数并不容易，尤其是后者，难道我新买一本书都要数一数字数吗？二是每次找书的时候，一般都是和书的内容有关，例如想找一本图论方面的研究生教材，查阅一个定理的文献出处，而不会想找一本二十一万一千零四十九个字的书。一般而言，连续的几次找书，可能出于类似的目的（找离散数学的相关教材和专著用于备课），或者相近的兴趣（这两月专攻沧月的小说及相似的玄幻作品），而不太可能一段时间内主要都找白色封皮的书或者按照从重到轻的顺序读书。

所以我想了又想，还是按照图书内容的分类来进行排列，包括小说、散文、诗歌、传记、历史、社会学、经济学、哲学、科普、科学专著、教材、工具书……同一类的图书，再按照作者姓名的字母序进行排列。后者倒是很好办，因为作者的名字是明确的，不会有误差。前者却不是图书固有的特征，而是经过某种非标准化的手段处理得到的，因为有很多个人的因素在里面——一本书可能属于很多不同的类别，既可以看成科学专著，也可以看成教材，取决于我自己的判断。不过不管怎样，这种符合使用习惯的分类，让我的书架变得非常清晰整齐，查书换书也非常方便。

因为把图书分多少类，怎么进行分类，都有个人的因素在里面，所以也会带来一些不便。我中学的时候有一个很好的朋友，曾经给我介绍过很多作者，既包括当时刚刚崛起的黄易的连载，也包括那个时候还不太出名的昆德拉和村上春树。现在快二十年过去了，我很想知道我们俩买书和看书的习惯还一样吗？遗憾的是，当你的书多到一定程度后，如果你没有办法直接记得哪些书你买过，要想计算有多少比例的书是重复的，就是件麻烦的事情——因为两个人对图书的分类和排列的方式不同，找起来特别费劲。这个时候，如果有一个人是按照页码进行排列，那就方便了，因为页码一翻就知道，而且一本书在甲手上是二百四十九页，到了乙手上不可能变成二百五十页。

自从开始打点自己的书柜，我租住的房子都变得更加干净了，也终于明白为什么需要采用中国图书分类号才能够管理一个图书馆。我真正梦想中的书柜，是按照图书的喜爱程度排序的。如果大家都这样排序，当我新遇到一个女孩子的时候，我就可以问她最喜欢的十本书，然后在我“心中的书柜”中查查这十本书的顺序。如果排名都比较靠前，那就可以继续交往了。这种方法可以避免我每次见到一个女孩子，总是忍不住询问“你是做什么方向的？”，“有没有简历可以给我简单看看？”等等尴尬的问题。

所以说，都按照中图分类号排列，是一种美；都按照自己心中喜欢的程度排列，也是一种美。两种美不知道哪个更美的时候，就不妨兼而得之。

周涛

电子科技大学互联网科学中心主任、教授、博士生导师

译者序



历时长达一年半，常心生退意，最终在众人的帮助下，本书的翻译工作总算落下帷幕。作为一名译者，我竭尽所能想呈现一部最好的作品，保留原作的精华和神韵，不过在翻译的过程中，因为专业知识受限及文化思维方式上的差异，难免遇到一些麻烦，庆幸的是我得到了许多从事数据库设计的专业人士的指导和帮助，在此我要对他们深表谢意。

作为一本数据库设计的指导性读物，读者看到书名的时候就当了解，本书所讲的并不是如何建立数据库的实际操作，重点落在设计两个字上。既然是设计，那就是一种概念性的思路，并不要求读者一定具备计算机方面的专业知识，所以本书不仅适用计算机专业人士，也能够成为所有数据库爱好者的床头读物。

本书层次分明，安排合理。作者为了让读者对书中的描述有更为具体的认识，从实际案例中取材，设置了案例分析部分。而让我记忆最深刻的是，第 5 章中作者介绍的如何开展访谈。与刻板的中国式教材不同，作者将自己丰富的经验和从亲身实践中总结出来的精华呈现了出来，因此文中的指南和意见非常人性化，可操作性很强。另外，作者创造性地发明了一些实用的方法和技巧，比如确定主题技巧、确定特征技巧。这些方法和技巧初看之下，会让人不以为然，但是到实际应用时，你会发现它的价值。

如果把本书比作一场盛宴，那第一部分则是上菜单。先介绍了数据库的类型、演变过程及发展方向。等到读者对关系数据库有了一个初步的了解之后，再来明确设计目标，紧接着又介绍术语。现在，不少读者求成心切，特

别是对于术语，往往没有耐心去记。作者把术语放在前头，好处就是读者看过一遍之后，不记得也没关系，大可以等到不懂的时候，再回过头去查找。这种记忆比起死记硬背来，确实更为有效。

第二部分就是本书的重头戏了。想必看过第一部分的读者，已经忍不住要开动了。作者显然明白读者们急切的心情，所以上来就让大家先试一下口，每样菜都尝一尝。这就是概念性概述。当然，读者朋友们不要挑食，要讲究营养均衡搭配才好。循着作者上菜的次序，一道一道品尝下来：建立表结构、确定主键、设置字段说明、建立表关系、确立业务规则、建立视图和各层次的数据完整性，你会发现原来每道菜都有讲究，缺一不可。

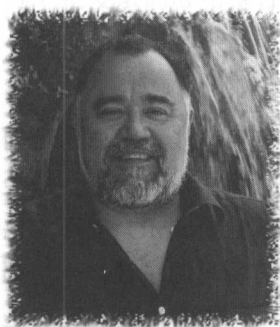
第三部分自然就是饭后甜点了。作者在这里对照前面规范的做法，介绍不规范的操作会带来的问题。所以，对于严格遵循作者前面的指导的读者，可以忽略这里不看。这就好比，前面吃饱、吃好了的食客无须再多吃。

吃完了，想必有人还想打包带回去吃。考虑周到的作者也想到了这一点。第四部分附录就完全是读者可以直接使用的材料，包括思考题答案、数据库设计过程中的示意图、整理好的设计指南、各种范表的模板、示意图符号、设计样板，等等。

我希望读者能够细细品尝这一桌丰盛的菜，等到你自己去做的时候，再回味这个味道，比较对照之下，就能有真正的收获。译文中有谬误之处，也请读者们不吝赐教，我们将不胜感激。

最后，要感谢所有参与了本书的翻译工作，并给予了我莫大帮助的亲朋好友——周涛、盛杨灿、李明、李果、肖琳、崔爱香、刘文才、钟蜜娴、胡猛、张伟、杨梦颖、胡飞、欧亚玲、张凡。

关于作者



Michael J. Hernandez 是一位独立的关系数据库咨询专家，专攻关系数据库设计。他有 20 多年的科技行业从业经验，为各种各样的客户开发过数据库应用程序。他为众多杂志专栏、白皮书、书籍和期刊撰稿，合著有畅销书 *SQL Queries for Mere Mortals*[®] (Addison-Wesley, 2007)。Mike (Michael 的昵称) 是全美政府、军队、私营部门以及企业中最受欢迎、最有名的技术培训师。他曾在很多美国国内和国际会议上发言，他的演讲和主持一直大受欢迎。

除了技术背景之外，Mike 的技能和兴趣广泛，从艺术到形而上学，涉猎众多。不过，Mike 最爱的还是吉他，他已经弹了 40 多年吉他，担任专业吉他手也有 15 个年头。Mike 厨艺了得，热爱教学（教授写作、公共演讲、音乐），擅长一语双关的文字游戏，甚至还能用塔罗牌算命。

Mike 说他永远不会真正地退休，只会在厌倦了手头上的事情之后，转而去做其他能引起他兴趣的事情。

编写这本书，作者必须有耐心去记。作者把本书放在书架上，并定期翻阅。经过一段时间，本书内容会逐渐清晰，直到可以写书的时候，再回过头去查找。本书的背景，确实更为有效。

序言

序言

本书是为那些对数据库设计感兴趣的人而写的。本书是一本入门书，也是一本参考书。本书的目的是帮助你了解数据库设计的基本原理，并为你提供实用的建议。本书的内容涵盖了数据库设计的全过程，从需求分析到数据库的实现。本书的风格简洁明了，易于理解。本书是一本值得一读的书。

致第 3 版

十年了，我和 Mike 见面的机会比过去更少了。有人可能不知道，我们同一天生日（虽然他比我年长了不少，起码大整整一岁）。我们每年至少会见上一面，庆祝我们又多活了一年。有趣的是，微软（Microsoft）差不多也是每十年就“更新”一次技术。如今，回头看十年前所写的序言，一切都没多大变化——我依然埋头研究微软的一项新技术，不过这一次研究的是 WinRT 和 Windows 8，而不是.NET。不过，有一点从未改变：人们对精心设计并良好执行的数据库设计的需求从未改变。Mike 在第 1 版中所述内容基本上依然适用，目前的新版本进一步完善了一些细节，而优良数据库设计的原则十年来始终如一。我不得不承认，我有点嫉妒 Mike 写了一本生命力如此持久的好书。一本书能畅销这么多年，至少证明它是一本好书。Mike 在本书中详细阐述了数据库设计过程，不管你是第一次阅读本书，还是第二次，抑或第三次，都肯定能在其中找到一条考虑周全的有益道路，指引你在变幻莫测的数据库设计中找到方向。现在，让我们跳过引言部分，进入正题吧！

——肯·盖茨（Ken Getz），2012 年 11 月 14 日

第2版序言

我和 Mike Hernandez 相见的次数不如从前多了。自我为本书的第一版作序以来，我们的职业生涯都发生了巨大的改变。就算不说别的，我们走动得少了，碰头的次数自然也就少了。如果你愿意迁就我，我也许会补充说，自此书的第1版问世以来，整个世界也发生了巨大变化。就最平凡无奇的方面来说，自我全心全意地投入到微软.NET 的开发中，我的整个开发生涯都被改变了。不过，有一件事始终未变，即对数据、对精心设计的数据的不断渴求。把复杂的应用整合在一起，但背后的数据却设计拙劣，由此带来的损害，不比 Mike 撰写本书第1版时的程度低。不管你是刚涉足数据库开发，还是已经经验丰富；不管你曾经读过 Mike 的书籍，还是第一次读到；不管你是乐于让别人设计你的数据，还是喜欢自己动手做——本书都是你的不二之选。Mike 不但能简单明了，而且能趣味横生地阐释概念，这一能力一直令我深感惊异。

——肯·盖茨 (Ken Getz)，2002 年 10 月 10 日

第1版序言

你可能好奇，为什么还要再出一本关于数据库设计的书呢？Mike Hernandez 第一次和我讨论这本书的时候，我也这么想。但事实就是，世上确实需要一本这样的书，也许通过迅速阅览本书的内容，在翻到本序言之前，你已经发现这一点。你肯定能找到许多解释数据库设计科学背后的原理和概念的书籍，但可能不会找到很多从 Mike 的独特视角出发的书籍（如果真能找到的话）。他立志拿出一本基于坚实的数学研究原理之上的书，但目标是实际应用，而不是讨论理论上的可能性。不管你现在具体使用的是哪种数据库，本书中的概念都有意义，适用于你的数据库设计项目。

当我翻到第6章的开篇，看到如下建议时，我就知道这本书正是我想要的：

不要将现有的数据库框架作为新的数据库框架的基础。

如果多年前，我刚刚踏上数据库开发者之路时，有人如此忠告我，我定能省下大笔的时间！这就是我所看中的：Mike 有多年为客户设计数据库的经验；他花费了大量时间思考、阅读、研究创造数据库应用的正确方式。在本书中，他毫无保留地将一切呈献给了大家。

这本书充满真知灼见，并用简单易懂的例子加以阐释。这并非是指本书中没有涉及适当的数据库设计的核心信息——书中自然是有这些信息的，但是这些内容是面向真正的程序开发人员的，而不是理论家。

我花了不少时间和 Mike 讨论数据库设计。不管是喝咖啡时、开会时还是写课件时，Mike 对数据库设计的热忱始终未减。就像操作系统设计师追求极致完美的算法一样，Mike 一直在探寻解决数据库设计的好方法，以及如何能恰当地解释给别人——就如你会在本书中看到的一样。多年来，我所掌握的数据库设计知识中，有很多是从 Mike 处学得，而且我确信此书定能教给我更多。本书简洁翔实地展示了开发专业数据库所需掌握的信息，我也确信你在读过这本书之后，会有同感。

——肯·盖茨 (Ken Getz)，MCW 技术公司 (KenG@mcwtech.com)

前言

生命，一如所有最古老的暗喻所认为的，是一段旅程。

——Jonathan Raban

For Love and Money

路线也许会变更，方向也许需要调整，但是旅途会继续……

——Micheal J. Hernandez

《数据库设计凡人入门》（第2版）

显然，从本书的第2版问世9年以来，科技领域，尤其是数据库管理方面的变化之巨，翻天覆地都不足以形容。如今，掌上设备的存储和处理能力，相当于过去几个房间大的大型机。今天，掌上设备无处不在，许多人视之为理所当然——尤其是年轻一代。（我的小侄子可能永远无法理解，我第一次为自己的IBM计算机买了一块40MB的扩展卡时的激动心情。不过，这又是另一回事了。）如今，数据库管理系统能够处理数TB的数据，现在人们也很重视存储、管理和利用云端的数据。

那么，现在是否还需要你手上拿的这本书？当然！无论数据库管理变得多么复杂，始终需要关于数据库设计基础的书籍。要理解事物运行的工作方式和原理，就必须掌握基础知识。许多其他的专业领域亦是如此，无论是技术类学科，如建筑设计、工程学，还是艺术类学科，如音乐、烹饪，无一例外。

近些年，我的人生踏上了新的、全然不同的道路，并且我非常享受所做的一切。最近，我写东西比较多，因此觉得是时候出本书的第3版了。我会分享在此过程中学到的一些新知识。也许，我还会进一步厘清我对这个主题的一些看法。现在我已经完成了这部作品，迫不及待地想知道这趟旅程接下来会带我到哪里。

读者请注意：参阅本书中引用的其他内容，请登录 Informit.com/titles/0321884493。

致谢

不管你听说过哪些写作心得，写作其实需要的是协作。我很感激一直乐于提供帮助的编辑、同事、朋友和家人一路上给我的支持。他们给我带来勇气，让我能专注于手头的工作，所以我想对他们致以最诚挚的谢意。

首先，我要感谢我的好编辑琼·默里（Joan Murray）给我重写一版的机会。我们花了好几年的时间讨论此事，是她的毅力、耐心、善良和领导力帮助我下定决心开始这项工作，并圆满完成。我还要感谢出版编辑卡罗琳·塞奈（Caroline Senay）熟练地指导审校，还有校对编辑奥德丽·多伊尔（Audrey Doyle）对本书内容精确、详尽的审校。同时，我要特别感谢约翰·富勒（John Fuller）以及他的制作人员——你们做得一如既往地出色！我一直和Addison-Wesley团队保持着良好的关系，我几乎不能想象我怎么会愿意为其他出版社写作技术书籍。

其次，我想感谢杰出的技术审校团队：特雷西·桑顿（Tracy Thornton）、托尼·威金斯（Tony Wiggins）还有西奥多·理查森（Theodor Richardson）。他们毫无保留地奉献出了他们的时间、精力和专业知识，给我提出了众多的宝贵意见和建议。毫无疑问，本书受益于他们的付出。我想再一次感谢你们每一个人的付出，让第3版的水平超出了我的预期。

非常感谢肯·盖茨（Ken Getz）再一次为本书作序，肯是一位备受尊敬的专家，同时也是我的同事、好友。真高兴能在本书开篇加上他的想法和评论。

我还要感谢读者朋友们把他们关于本书的想法和评论反馈给我。大家给予的赞扬和支持让我受宠若惊，我要特别感谢读者提出的建设性批评意见，让我得以在新一版中完善相关内容。我还要感谢所有的学术机构、政府组织和商业组织，感谢你们采用我的作品，当作数据库初学者的“标准读物”。它们对本书的肯定，让我备感荣幸。

最后，我想感谢我的妻子，感谢她在我创作此书的过程中给予我无尽的耐心。她的帮助和支持何其珍贵，我欠她的太多。我可以清楚地告诉大家我有多爱她，但是她讨厌任何公开示爱的行为。因此，我只能为她献上桂冠，紧紧地握住她的手。

引言



平凡的厨子做不好普通的菜。

——Countess Morphy

过去，数据库设计的工作由信息技术（IT）部门人员和专业的数据库开发者履行。他们通常具备数学、计算机科学或者系统设计背景，往往有使用大型机数据库的经验。其中很多人编程经验丰富，已经编写过大量数千行代码的数据库应用程序。（他们因其工作的性质和重要性，通常都会工作过度。）

那时，设计数据库系统的人员必须有过硬的学历背景，因为他们开发的大部分系统都将在全公司范围内运用。由于编程语言和他们所采用的数据库应用程序异常复杂，即便只是为公司内部的一个部门或者一个小公司开发数据库，数据库设计师仍然需要进行全面的正规训练。然而，随着技术进步，教育背景要求也在变化。

自 20 世纪 80 年代以来，数据库软件取得了长足发展。许多供应商都开发出了能运行在台式机上的软件，相比大型机软件，这些产品收集、存储和管理数据更见简易。随着计算处理能力的增强，以及对复杂度的要求提升，供应商还开发出了能在各种不同环境下让许多人访问和共享集中数据的软件，例如在局域网（LAN）和广域网（WAN）连接的计算机上运行的用户/服务器架构。公司或组织机构内部人员不再严格依赖大型机数据库或由中央 IT 部门来满足他们的信息需求。

笔记本电脑的出现和广泛应用，以及互联网的普及也推动了数据库软件

的发展。笔记本电脑已经变得非常强大，拥有数 GB 的内存和存储容量以及飞速的处理能力。在很多地方，笔记本电脑已经非常普及，甚至完全取代了台式电脑。这也使得人们在咖啡屋、餐馆和机场等常见的地方能够连上互联网。（我甚至无须提及各色能连上互联网的设备——这当有别的书来另行讨论。）如此，软件供应商和企业双双推动在互联网上运营和管理数据库，以让更多的人能够随时随地访问到他们的应用程序和数据。看看未来几年这种想法的进展如何，将十分有趣。

供应商不断地在他们的数据库软件中添加新功能，增强工具集，这使得数据库开发人员能够创建更强大、更灵活的数据库应用。他们也在不断地提升软件的易用性，使更多人能够创建自己的数据库应用。当下的数据库软件大大简化了创建高效的数据库结构和直观的用户界面的过程。

大部分程序都提供示例数据库结构，让用户根据自身的特定需求来复制或修改。虽然用户一开始可能认为，以这些示例结构为基础建立新数据库十分方便，但现在你应当停下来，重新思考一下这一举动。为什么？因为你有可能在不知不觉中创建出一个不当、低效、不完整的设计。最终可能会在一个自认为可信赖的数据库设计上遇到问题。当然，这就提出了一个问题：“我会遇到什么类型的问题呢？”

数据库中出现的大部分问题可以归为下面两类：应用问题和数据问题。应用问题包括数据输入/编辑表单有问题、令人困惑的菜单和工具栏、令人难以理解的对话框和烦琐的任务顺序。这些问题主要是因为数据库开发人员经验不足、不熟悉良好的应用设计方法或者不了解用来实现数据库的软件。这类问题很常见，妥善解决这类问题很重要，不过，这不在本书的讨论范围之内。

❖注意：解决遇到的许多应用问题的一个好方法就是，购买并研读一本第三方“开发者”写的涉及你采用的软件的书。这类书籍探讨了应用设计问题、高级编程技术以及可以用来改善优化一个应用的各种技巧和窍门。具备了这些新技能，就能改善和微调数据库应用程序，使之能正确、平稳和有效地运行。

另一方面，数据问题包括诸如数据缺失、数据不正确、数据不匹配以及信息不精准。糟糕的数据库设计通常是造成这类问题的根源。如果一个数据库结构不合理，就没法满足一个组织机构的信息需求。虽然出现糟糕的设计通常是由于数据库开发者未掌握良好的数据库设计原则，但是这并不能表明开发者水平低。很多人，包括经验丰富的程序员和数据库开发人员，很少或从未接受过任何形式的数据库设计方法方面的指导。很多人甚至从未意识到存在设计方法。数据问题和糟糕的设计是本书将解决的问题。

第3版中的新变化

笔者修订了这一版，提升了可读性，更新和扩展了已有内容，增加了新内容，提高了本书的教学价值。第3版中的一些新变化如下：

- 重写了部分文字，使意思更加清晰，便于读者理解。
- 修订了图表内容，增强实用性。
- 更新了对数据类型的讨论。
- 推荐阅读书目中涵盖了书籍的最新版本和各书的国际标准书号（ISBN）。
- 增加了关于规范化的附录。这一部分简明地解释了规范化这一概念，以及如何融入到本书所讨论的设计过程之中。

参阅本书中引用的其他内容，请访问网址：informit.com/titles/0321884493。

读者群体

阅读本书，不需要拥有数据库设计背景。因为读者拿起本书，就是为了学习如何恰当地设计数据库。如果你初涉数据库管理领域，正在考虑开发一个自己的数据库，那本书就有价值。从一开始就掌握正确的数据库创建方法，比通过反复试验试错来学习要好得多。相信我，后者需要的时间长得多。

如果你和数据库编程打过一段时间的交道了，正打算为公司或企业开发新数据库，那么应该读一读本书。也许你心里大致有数，优良的数据库结构

应该是个什么样子，但不是很确定数据库开发人员要如何实现有效的设计。也许你是一个程序员，按照一些基本原则，已经创建过不少的数据库，但是总免不了要编写大量的程序代码，才能让数据库正常运行。如果你属于这种情况，那么本书也值得一读。

即使你已经有了数据库设计方面的背景，最好也能读一读本书。也许你大学时学过数据库设计方法，或者上过数据库相关课程，课上讨论过数据库设计，但已记不清细节，或没有理解某些设计过程。不过，只要学习并理解了本书中讲述的设计过程，那些曾经困惑之处都将豁然开朗。

本书也适用于经验丰富的数据库开发人员和程序员。虽然你可能已经了解了书中讲述的设计过程的诸多方面，但是读下去就可能发现一些以前从未遇到或考虑过的问题。通过阅读此书，可能受到启发，产生一些关于如何设计数据库的新想法，因为本书从不同的角度阐释了许多你所熟知的设计过程。至少，本书可当作一本不错的数据库设计复习教程。

本书的目的

总体来说，整个数据库开发过程分为三个阶段。

1. **逻辑设计**：涉及定义表及其字段、建立主键和外键、建立表之间的关系，并确定和建立各级数据完整性。
2. **物理实现**：涉及建表、建立主键字段和表之间的关系，并使用恰当的工具来实现各级数据完整性。
3. **应用开发**：涉及创建一个应用，能让单个用户或群组用户利用存储在数据库中的数据。此阶段本身又可划分为多个独立的步骤，譬如确定终端用户任务和适当的序列、确定输出报表的信息需求，以及为应用导航创建一个菜单系统。

你自始至终应该先从逻辑设计阶段着手，并尽可能彻底地执行所做的设计。在建立了健全的结构之后，就能应用到所选的任何数据库软件中了。在

物理实现阶段，可能会需要根据自身所选软件的利弊和优劣，适当调整数据库结构。甚至可能会为了提升数据处理性能而调整数据库结构。首先进行逻辑设计可确保在确定数据库结构时，做出有意识的、有条理的、清晰明智的决策。如此一来，也有助于减少在物理实现和应用开发阶段进一步调整数据库结构的次数。

本书只涉及整体开发过程中的逻辑设计阶段。本书的重点是，在摒弃绝大部分数据库设计书籍采用的高深、正统的方法论的前提下，阐释关系数据库的设计过程。笔者尽量避开这些复杂的方法论，转而呈现一种相对简单明白的设计方法。笔者也采用了一种简单直接的数据建模方法作为此设计方法的辅助，以尽可能清晰的语言、尽量少的技术术语，讲述整个设计过程。

市面上的许多数据库设计相关书籍都包含在特定的数据库产品中实现数据库的章节。有些书籍甚至会将设计和实现阶段混为一体。（笔者非常不赞成这种做法，而且一直坚持认为一个数据库开发者应当将逻辑设计和实现阶段分开，以确保最大程度的集中、高效。）这类书籍的主要缺陷是，如果读者没有使用书中提到的特定数据库软件或编程语言，就很难从有关数据库实现的章节中获得任何有用或相关的信息。因此，笔者才决定编写一本专注于数据库逻辑设计的书籍。

相比同一主题的其他书籍，本书可能更易读。市面上的许多数据库设计相关书籍技术性很强，可能难于理解吸收。笔者认为，如果没有读过计算机科学专业，不是一个数据库理论家或者是有经验的数据库开发人员，这些书绝大部分让人困惑，望而生却。本书中的设计原则易于理解、记忆，书中所采用的示例也非常常见，具有通用性，适用于各种情况。

笔者在全美巡讲的时候，碰到的大部分人都表示，他们想学习创建合理的数据库结构的方法，而不用掌握范式和高等数学理论。相比在特定数据库软件应用程序中实现一种结构，很多人更关注如何优化他们的数据库结构和实施数据完整性。在本书中将学习如何创建高效的数据库结构、如何实施多层数据完整性，以及如何用几近无限量的方式将表关联在一起以获取信息。不必惊慌，这没你想象的那么难。通过了解一些关键术语、学习和使用一组

特定的常见技巧和概念，就能做到这一切。

同时，读者将学习如何分析和利用现有数据库、确定信息需求，以及确定和实施业务规则。这些话题都很重要，因为许多人也许会沿用旧的数据库，需要利用通过阅读此书所学到的东西来改进。不过，就算从头开始创建一个数据库，这些话题照样很重要。

读完本书，也就掌握了创建一个优良的关系数据库结构所必不可少的知识和工具。笔者坚信，这一整套方法将对多数数据库开发人员和他们要创建的数据库都有效。

如何阅读本书

不管你是新手还是专业人士，强烈推荐按顺序从头读到尾。如此一来，就不会脱离上下文语境，避免困惑——疑惑通常是由于一开始没能看到本书的全局造成的。此外，在重点研读某个章节之前，先整体了解整个过程，不失为一个好主意。

当然，如果读本书是为了复习设计技巧，那么也可以只读那些感兴趣的章节。虽然笔者已经尽可能地让每一章能独立成篇，但仍然推荐读者能将整本书通读一遍，以确保不会错过任何至今为止你从未想到过的、有关数据库设计的新思想或新点子。

本书的组织结构

下面是本书各个部分和各个章节所述内容的简单概述。

第1部分：关系数据库设计

该部分介绍数据库、数据库设计的基本思想，以及一些为了学习和理解本书所讲述的设计过程所需熟知的术语。

第1章 关系数据库

本章简单地讨论了数据库类型、常见的数据库模型和关系数据库的发展简史。

第2章 设计目标

本章探讨了需要关注设计的原因，指出了优良设计的目标和优点，并简单介绍了规范化和范式。

第3章 术语

本章涵盖了为了学习和理解本书中所讲述设计方法所必须掌握的术语。

第2部分：设计过程

该部分详细论述了数据库设计过程的方方面面，包括创建表结构、指定主键、设立字段说明、建立表间关系，以及建立视图和各种不同层次的数据完整性。

第4章 概念性概述

本章概述了设计过程，阐释了设计过程的各个部分是如何衔接在一起的。

第5章 大幕开启

本章介绍了如何定义数据库的宗旨和任务目标，并且两者都是数据库创建初始阶段的重心。

第6章 分析现有数据库

本章涵盖了现行数据库存在的问题。探讨了为什么要分析现行数据库、如何看待收集和呈现数据的现有方法、为何和如何与用户和管理人员交谈，以及如何编写初始字段列表。

第7章 建立表结构

本章涉及的内容包括诸如界定数据库该记录什么内容，将字段与表联系

起来，以及精简表结构。

第 8 章 键

本章涉及内容包括键的概念以及其对设计过程的重要性，还有如何为每个表定义候选键和主键。

第 9 章 字段说明

本章探讨了一个众多数据库开发人员容易忽略的问题。字段说明除了指示每个字段如何创建之外，还决定了每个字段取值的特性。本章涉及的内容包括字段说明的重要性、说明特性的类型和如何定义数据库中每个字段的说明。

第 10 章 表关系

本章阐释了表间关系的重要性、关系类型、设立关系和建立关系特征。

第 11 章 业务规则

本章讲述了业务规则的类型、确定和设立业务规则，以及使用验证表。业务规则对任何数据库都非常重要，原因在于它决定了一个方面的数据完整性。

第 12 章 视图

本章探讨了视图的概念及其重要的原因，介绍了视图类型以及如何确定和建立视图。

第 13 章 评审数据完整性

本章对前面章节所定义和讨论过的各个层次的数据完整性做了评审。读者应该认识到，评审数据库结构的最终设计，能有效确保数据完整性尽可能完善。

第3部分：其他数据库设计事项

该部分讨论了在设计过程中如何避免设计不当以及打破规则的问题。

第14章 设计不当——禁忌事项

本章涵盖了应避免出现的设计问题类型，比如平面文件设计和电子表格设计。

第15章 打破规则

本章讨论了一些需要对设计过程的技巧和概念进行适当调整的特例，介绍了何种情况下应考虑放宽规则，以及如何进行调整。

第4部分：附录

附录部分提供了一些对了解数据库过程和开发数据库时有用的信息。

附录A 思考题答案

本部分中包含了从第1章到第12章所有思考题的答案。

附录B 数据库设计过程中的示意图

本部分提供了整个设计过程所使用的示意图。

附录C 设计指南

本部分列出了本书中出现的各种设计指南，以便读者查阅。

附录D 文档形式

本部分提供了字段说明、业务规则规范、视图规范表的空白表格，以便读者使用。

附录E 数据库设计示意图符号

本部分包含了本书中出现过的所有示意图符号，以便读者查阅。

附录 F 设计样本

本部分介绍了一些数据库设计样本，读者可以从中借鉴一二。

附录 G 关于规范化

本部分介绍了作者如何将规范化融入本书的设计方法中。

附录 H 推荐书目

本部分列出了读者深入学习数据库技巧所应阅读的一些书籍。

术语表

本部分为本书中用到的所有术语给出了简明的定义。

❖注意：务必阅读本部分！

关于本书中的示例和技巧

本书中运用了大量示例。笔者在选择示例时，也尽量确保它们普遍通用、切中主题。不过，读者难免还是会发现一些示例过于简单、不够完整甚至有时是错误的。无论读者相信与否，笔者确实是有意为之。

之所以运用了一些存在错误的示例，是为了将特定概念或技巧阐述清楚。缺少这些示例，读者就无法理解如何使用对应概念或技巧，也无法看到预料中的结果。一些示例过于简单，同样是因为重点在于对应技巧或概念，而非示例本身。例如，设计一个跟踪订单的数据库有很多种方法。但是，本书中使用的订单跟踪数据库示例的结构却比较简单，原因是重点仅在于设计过程，而不是建立一个详尽的订单跟踪数据库系统。

所以，笔者在此力图真正强调的是：

重点在于概念或技巧及其预期结果，而非所使用的示例。

新的学习方法

下面介绍一种学习设计过程（或者任何相关的知识）的非常有用的新方法，这是笔者在自己开设的数据库设计课上发现的。

把设计过程中用到的所有技巧当作一套工具，每个工具（或技巧）都有特定的用途。这里的想法就是，一旦掌握了一个工具的一般用途，就可以在任意情况下使用该工具。原因在于，每种情况下使用该工具的方式都是一样的。

就拿活动扳手来说。一般说来，活动扳手用来拧开或拧紧螺钉。调整扳手头部的调校螺丝，让活动扳手的开口尺寸套上给定的螺母。现在，既然已经清楚了它的用法，就找几个螺钉尝试一下。比如，户外椅的椅腿、发动机的风扇皮带罩、空调室外机的侧板或铁门的铰链板。是否发现，无论在哪儿遇到螺栓和螺母，始终能用相同的方法紧固和拧开螺母？

设计数据库所用的工具也是如此。一旦懂得了一个工具的一般用法，无论遇到怎样的情况，都能奏效。例如，想想分解字段值的工具（或技巧）。在 CUSTOMERS 表中有一个 Address 值，这个表包含了特定顾客的道路地址、城市、州以及邮编的信息。你会发现很难在数据库中使用这个字段，因为它包含了多个数据项。无疑，检索特定城市的信息或通过特定邮编对信息分类会非常困难。

解决这一难题的方法是将 Address 字段分解成更小的字段。先识别组成该字段值的各个项，再将每个项当作一个独立的字段，问题就迎刃而解了。就是这么简单！这个过程中包含了一个“工具”，现在可以将它应用到任何包含多个不同数据项的字段中，比如下面的字段样本。下列表中展示了分解过程的结果。

当前字段名	示例值	新字段名
Address	7402 Kingman Dr.,Seattle. WA 98012	Street Address, City, State, Zip Code
Phone	(206)555-5555	Area Code, Phone Number
Name	Michael J. Hernandez	First Name, Middle Initial, Last Name
EmployeeCode	ITDEV0516	Department, Category, ID Number

❖注意：第 7 章“建立表结构”中对分解字段值做了更为详细的介绍。

可以用相同的方式使用本书设计过程中呈现的所有技巧（“工具”）。无论需要创建什么类型的数据库，使用这些技巧，都能设计出完善的数据库结构。但是要切记：

重点在于概念或技巧及其预期结果，而非所使用的示例。

目录

第 1 部分 关系数据库设计.....	1
第 1 章 关系数据库.....	2
本章内容	2
数据库的类型	3
早期数据库模型	4
层次数据库模型	4
网状数据库模型	7
关系数据库模型	10
检索数据	12
关系数据库的优势	13
关系数据库管理系统	14
后关系模型	16
未来将会如何	17
最后一点	17
小结	18
思考题	19

第 2 章 设计目标	20
本章内容	20
为什么要关注数据库设计	21
理论的重要性	22
学习优秀设计方法学的益处	23
优秀设计的目标	24
优秀设计的好处	25
数据库设计方法	25
传统设计方法	25
本书中所呈现的设计方法	27
规范化	27
小结	29
思考题	30
第 3 章 术语	31
本章内容	31
术语为何重要	32
关于值的术语	33
数据	33
信息	33
空值 (NULL)	34
NULL 的值	35
NULL 所带来的问题	36
关于结构的术语	38
表	38

字段	40
记录	41
视图	42
键	44
索引	45
关于关系的术语	46
关系	46
关系类型	47
一对一关系	47
一对多关系	48
多对多关系	49
参与的类型	51
参与度	52
关于完整性的术语	53
字段说明	53
数据完整性	53
小结	54
思考题	55
第 2 部分 设计过程.....	57
第 4 章 概念性概述.....	58
本章内容	58
完成设计过程的重要性	59
明确宗旨和任务目标	60
分析现有数据库	60

创建数据结构	61
确定和建立表关系	62
确定和定义业务规则	63
确定和定义视图	63
审核数据完整性	64
小结	65
思考题	66
第 5 章 大幕开启	67
本章内容	67
开展访谈	68
参与者指南	69
采访者指南	70
案例分析：迈克自行车行	73
明确宗旨	74
优良的宗旨	74
制订宗旨	75
案例分析：为迈克自行车行制订宗旨	77
明确任务目标	78
优秀的任务目标	78
制订任务目标	79
案例分析：为迈克自行车行制订任务目标	82
小结	83
思考题	83

第 6 章 分析现有数据库.....	85
本章内容	85
了解现有数据库	85
纸质数据库	87
遗留数据库	88
开展分析	89
了解如何收集数据	89
了解如何呈现信息	92
开展访谈	95
基本访谈技巧	96
开始访谈之前	100
用户访谈	101
评审数据类型和用途	101
评审样本	102
评审信息要求	106
管理人员访谈	112
评审当前信息要求	112
评审附加信息要求	113
评审未来信息要求	113
评审总体信息要求	114
编辑完整字段列表	115
初始字段列表	115
计算字段列表	120
案例分析	121
小结	126

28	思考题	126
第 7 章 建立表结构.....		128
28	本章内容	128
28	定义初始表列表	129
28	确定隐含主题	129
28	使用主题列表	130
28	使用任务目标	134
28	定义最终表列表	136
28	改进表名称	137
28	指明表类型	140
001	编辑表描述	141
101	字段对应入表	145
101	精简字段	147
201	改进字段名称	147
301	使用理想字段解决异常现象	150
411	消除复合字段	152
411	消除多值字段	154
411	精简表结构	159
411	谈谈冗余数据和重复字段	159
411	参照理想表精简表结构	160
411	建立子集表	166
411	案例分析	170
051	小结	175
151	思考题	176

第 8 章 键	178
本章内容	178
键为何重要	179
为每个表建立键	179
候选键	179
主键	185
替换键	190
非键	191
表层次完整性	191
评审初始表结构	191
案例分析	192
小结	197
思考题	198
第 9 章 字段说明	199
本章内容	199
字段说明为何重要	200
字段级完整性	201
字段说明之剖析	202
一般元素	204
物理元素	209
逻辑元素	213
使用独特、通用和可复制的字段说明	219
定义每个字段的字段说明	225
案例分析	226

小结	228
思考题	229
第 10 章 表关系	231
本章内容	231
关系为何重要	232
关系的类型	233
一对一关系	234
一对多关系	235
多对多关系	237
自联结关系	244
识别现有关系	247
建立关系	255
一对一和一对多关系	255
多对多关系	260
自引用关系	265
评审表结构	269
改进所有外键	270
外键的要素	270
建立关系特征	275
为每个关系定义删除规则	275
识别每个表的参与类型	279
识别每个表的参与度	281
与用户和管理人员验证表关系	283
结语	283
关系层次完整性	284

案例分析	284
小结	289
思考题	290
第 11 章 业务规则	292
本章内容	292
什么是业务规则	293
业务规则类型	295
业务规则的分类	297
字段特有业务规则	297
关系特有业务规则	298
定义和建立业务规则	299
与用户和管理人员合作	299
定义和建立字段特有业务规则	300
定义和建立关系特有业务规则	307
验证表	312
什么是验证表	313
使用验证表支持业务规则	313
评审业务规则规范表	318
案例分析	318
小结	324
思考题	325
第 12 章 视图	326
本章内容	326
什么是视图	326

视图之剖析	328
数据视图	328
聚合视图	332
验证视图	335
确立视图	336
与用户和管理人员合作	337
定义视图	337
评审每个视图的文档记录	345
案例分析	345
小结	350
思考题	351
第 13 章 评审数据完整性	352
本章内容	352
为什么要评审数据完整性	353
评审和改进数据完整性	353
表层次完整性	354
字段级完整性	354
关系层次完整性	354
业务规则	355
视图	355
汇编数据库文档	356
大功告成	357
案例分析——总结	357
小结	357

第 3 部分 其他数据库设计事项.....359

第 14 章 设计不当——禁忌事项..... 360

本章内容	360
平面文件设计	361
电子表格设计	362
摒弃电子表格视图思维定式	363
基于数据库软件设计数据库	364
最后一点想法	365
小结	366

第 15 章 打破规则

本章内容	367
何种情况下可以打破规则	367
设计分析型数据库	368
提升处理性能	368
提升性能首选其他方式	369
记录行动	370
小结	371

结束语..... 372

第 4 部分 附录.....375

附录 A 思考题答案

附录 B 数据库设计过程中的示意图

附录 C 设计指南..... 410

附录 D 文档形式	418
附录 E 数据库设计示意图符号	422
附录 F 设计样本	424
附录 G 关于规范化	431
附录 H 推荐书目	438
术语表	440
参考文献	453

第1章

第 1 部分

关系数据库设计

容内章本



第 1 章

关系数据库

鱼，要想品味正，应游泳三次——在水里游、在油里游和在酒中游。

——波兰谚语

本章内容

数据库的类型

早期数据库模型

关系数据库模型

关系数据库管理系统

后关系模型

未来将会如何

小结

思考题

关系数据库 (relational database) 诞生至今已 40 余年。它不仅催生了一个价值数十亿美元的产业，也是当今世界使用最为广泛的数据库类型，成为我们日常生活中必不可少的一部分。无论你是在网上商城还是当地的商场购物，与旅行社制订旅行计划还是从图书馆借阅图书，你都很有可能是在使用关系数据库。

在深入研究关系数据库的设计过程之前，不妨来了解一下它的简史——它的起源、现状以及未来发展方向。

数据库的类型

什么是数据库？你可能知道，数据库是一种组织化的数据集合，目的是为某种类型的组织或组织过程建模。事实上，不管你是使用什么手段收集和存储数据，通过纸笔还是计算机应用程序，都无关紧要。只要是为了某一特定的目的，并且以一定组织化的方式收集数据，数据库就已经产生了。在接下来的讨论中，我们将假定使用应用程序收集和维持数据。

在数据库的管理中有两种数据库，操作型数据库（operational database）和分析型数据库（analytical database）。

操作型数据库是世界各地许多公司、组织和机构的支柱。这种数据库主要用于联机事务处理（online transaction processing, OLTP），即需要收集、修改和维护日常数据的情况。操作型数据库中存储的数据呈动态，意味着它在不断变化且始终反映即时信息。诸如零售店、制造企业、医院和诊所，以及出版社之类的机构都会使用操作型数据库，因为它们的数据总是处于不断更新的状态。

相比之下，分析型数据库主要用于联机分析处理（online analytical processing, OLAP），用来存储并追踪历史性和时间性的数据。如果需要跟踪趋势，查看长时期的统计数据，并制订战略性或战术性业务预测，分析型数据库就会是一笔宝贵的财富。这种数据库存储静态数据，意味着数据永远不会（或极少）被修改。从分析型数据库中得到的信息反映的是某一时间点的数据库简况。使用分析型数据库的机构包括化学实验室、地质公司以及市场分析公司等。

分析型数据库往往将操作型数据库中的数据作为其主要数据源，因此它们之间有一定的联系；不过，操作型和分析型数据库满足不同的数据处理需求，各自创建的方法也完全不同。本书主要介绍如何设计操作型数据库，因

为它仍然是当今世界上最常用的数据库类型。

早期数据库模型

在关系数据库模型出现之前，有两种通用数据模型用于维护和操作数据：层次数据库模型（hierarchical database model）和网状数据库模型（network database model）。

❖注意：本书仅从历史角度提供每个模型的简要概述。整体而言，笔者认为了解关系数据库模型之前的历史对读者有利——可以对关系数据库因何产生并演变有一个基本的认识。

在下面的概述中，将简要介绍每个模型中的数据是如何构造并访问的，两个表之间的关系是如何体现的，以及每种模型的一些优缺点。

本节中出现的一些术语在第3章“术语”中做了更为详尽的描述。

层次数据库模型

这种数据库中的数据是按照层次结构组织的，通常图解为倒置的树状结构。其中一个表作为倒置树状图的“根”，其他表则是由根生发的枝条。图1.1是一个典型的层次数据库的结构图。

经纪公司数据库

如图1.1所示，一个经纪人（agent）签约若干艺人（entertainer），而每位艺人都有各自的行程安排表（schedule）。同时，经纪人通过满足客户的娱乐需求（engagement），维持有一定数量的客户（client）。然后，客户通过经纪人签订服务合同，同时将服务费用支付给经纪人。

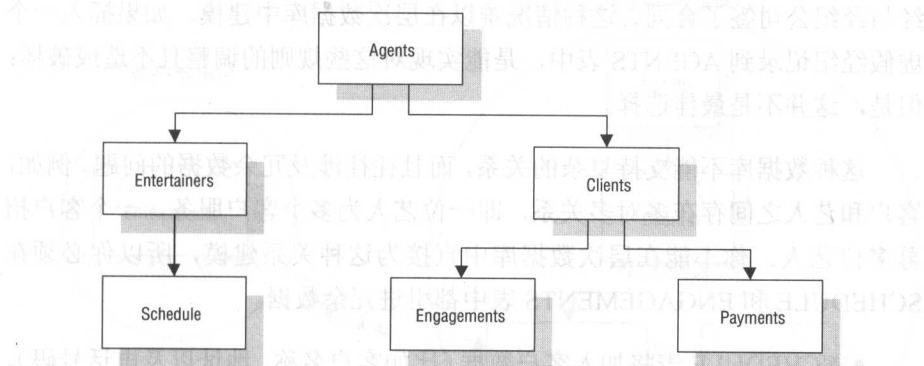


图 1.1 典型的层次数据库示意图

层次数据库中的关系由术语父/子（parent/child）代表。在这种关系中，父表（parent table）可以与一个或多个子表（child table）相连，而子表却只能和一个父表相连。这些表通过一个指针或表内记录的物理排列建立明确的联系。用户访问这个模型中数据的方式就是，从根表开始，一直沿着树状结构到达目标数据。这种访问方式要求用户对数据库的结构非常熟悉。

使用层次数据库的一个优点是，用户可以迅速检索到数据，因为表结构之间有明确的联系。另一个优点则是内嵌了参照完整性（referential integrity），并且是自动执行的。这就确保了子表中的记录必须与父表中的已有记录具有联系，父表中删除一个记录就将造成子表中所有相关记录也随之被删除。

❖注意：在接下来的示例中，文中的表名称均以大写字母的形式出现（比如 VENDORS），字段名以首字母大写的形式出现（比如 Vendor ID Number）。

但是，当用户需要在子表中存储一个记录，而该记录与父表中的任何记录都没有联系时，就会出现这个问题。考虑一下使用图 1.1 所示的经纪公司数据库。用户不能在 ENTERTAINERS 表中添加一位新艺人，除非该艺人被分派给了 AGENTS 表中的一位经纪人。记住，子表中的一个记录（这个例子中是指 ENTERTAINERS）必须与对应父表（AGENTS）中的记录产生联系。但是在现实生活中，通常情况是，早在艺人被指定给特定经纪人之前，就已

经与经纪公司签了合同。这种情况难以在层次数据库中建模。如果插入一个虚假经纪记录到 AGENTS 表中, 是能实现对这些规则的调整且不造成破坏; 但是, 这并不是最佳选择。

这种数据库不能支持复杂的关系, 而且往往涉及冗余数据的问题。例如, 客户和艺人之间存在多对多关系, 即一位艺人为多个客户服务, 一个客户招募多位艺人。你不能在层次数据库中直接为这种关系建模, 所以你必须要在 SCHEDULE 和 ENGAGEMENTS 表中都引进冗余数据。

- SCHEDULE 表将加入客户数据 (比如客户名称、地址以及电话号码), 用以表示每位艺人的服务对象和地点。这种数据是冗余数据, 因为它当前存储在 CLIENTS 表中。
- ENGAGEMENTS 表将包含艺人数据 (比如艺人姓名、电话号码以及艺人类型), 以表明给定客户应选择哪些艺人。这种数据也是冗余数据, 因为它当前存储在 ENGAGEMENTS 表中。

这种冗余数据造成的一个问题就是, 用户可能可以输入不同数据。另一方面, 这可能导致生成的信息不准确。

用户可以采取一种迂回的方式解决这个问题, 即为艺人和经纪公司分别建立一个层次数据库。新型艺人数据库将只包含 ENTERTAINERS 表, 改进的经纪公司数据库将包含 AGENTS、CLIENTS、PAYMENTS 以及 ENGAGEMENTS 表。艺人数据库不再需要 SCHEDULE 表, 因为你可以在经纪公司数据库中的 ENGAGEMENTS 表和艺人数据库中的 ENTERTAINERS 表之间定义一个逻辑子关系。借助这种关系, 就可以检索各种信息, 比如为特定客户提供签约艺人名单或者查看特定艺人的行程安排表。图 1.2 就是新模型的示意图。

从中可以看出, 设计层次数据库的开发人员必须认识到采用这种技术来表示多对多关系的必要性。在这里, 这种必要性相对明显, 但是还有许多关系会更为隐晦, 需要等到设计过程后期, 更有甚者需要在数据库投入使用很长一段时间后, 才能被发现。

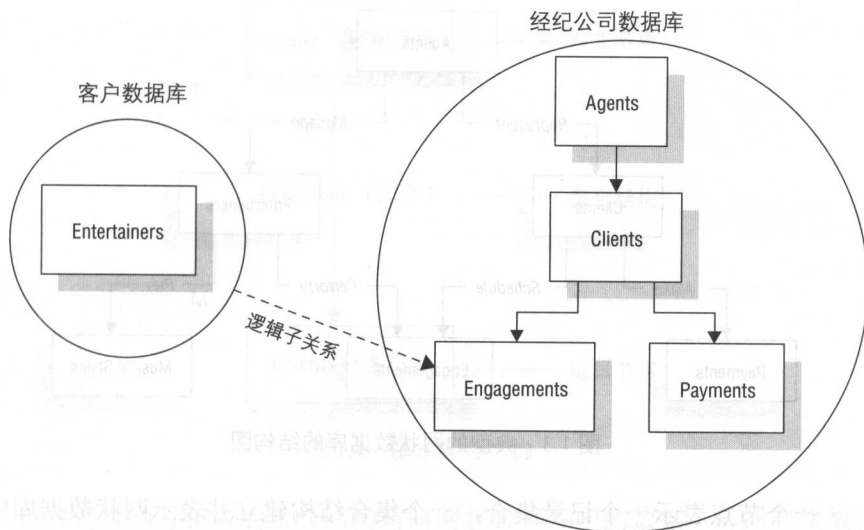


图 1.2 使用两个层次数据库解决多对多关系

层次数据库非常适用于 20 世纪 90 年代大型机所使用的磁带存储系统 (tape storage system)，并且在使用这类系统的机构中也十分受欢迎。不过，尽管层次数据库能够快速直接地访问数据并且在多数情况下都有效，但是显然，我们还是亟需一种新型数据库模型来解决日益严峻的数据冗余和数据间的复杂关系这些相关问题。

网状数据库模型

在很大程度上，网状数据库的开发就是为了解决层次数据库出现的一些问题。网状数据库的结构用术语节点 (node) 和集合结构 (set structure) 表示。图 1.3 就是一个典型的网状数据库的示意图。

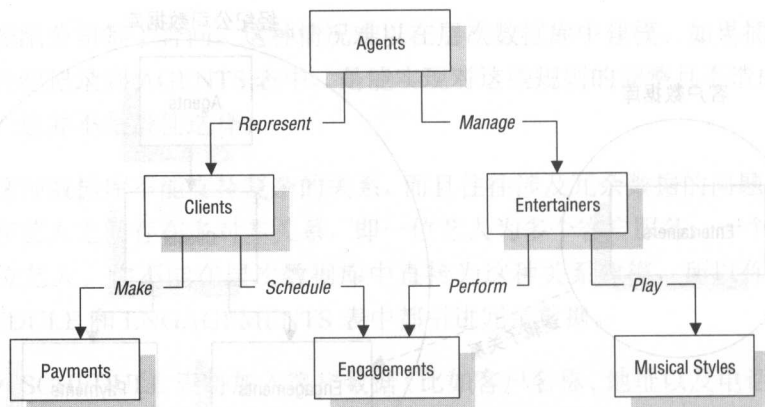


图 1.3 典型的网状数据库的结构图

一个节点表示一个记录集合，一个集合结构建立并表示网状数据库中的一种关系。这种结构条理清晰，使用一个节点作为系主（owner），另一个节点作为成员（member），从而将一对节点联系起来。（相对父/子关系，这是一个不错的改进。）集合结构支持一对多关系，也就意味着系主节点中的一个记录可以与成员节点中的多个记录产生联系，不过成员节点中的单个记录只与系主节点中的一个记录相关。另外，成员节点中的一个记录必须与系主节点中的一个现有记录有联系，否则该记录不存在。例如，一个客户必须指定给一个经纪公司，但是没有客户的经纪公司仍然可以在数据库中列出来。图 1.4 为一个基本集合结构的示意图。

一对节点之间可以定义一个或多个集合（联系），单一节点也可以与其他节点形成其他集合。例如，在图 1.3 中，CLIENTS 节点通过 Make（支付）集合结构与 PAYMENTS 节点产生联系，也通过 Schedule（安排）集合结构与 ENGAGEMENTS 节点产生联系。在与 CLIENTS 节点联系的同时，ENGAGEMENTS 节点也通过 Perform（执行）集合结构与 ENTERTAINERS 产生联系。

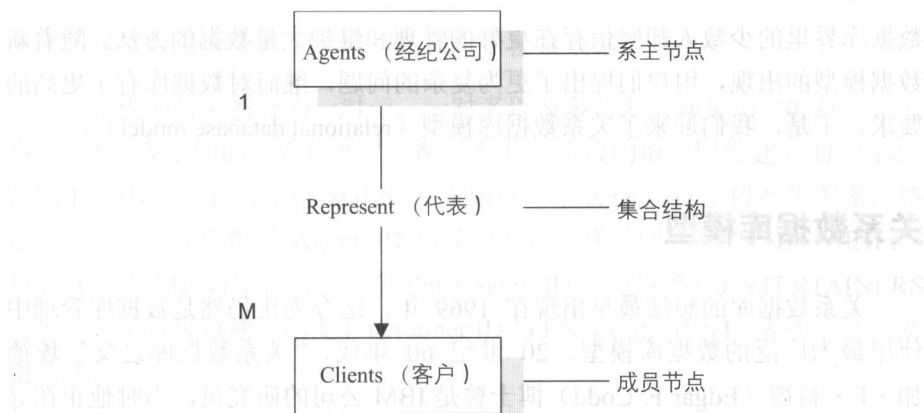


图 1.4 基本集合结构

用户可以通过相应的集合结构访问网状数据库中的数据。访问层次数据库必须从根表开始，而网状数据库却不同，用户可以在网状数据库内部访问数据：从任意节点开始，沿着相关集合正向或反向访问网状数据库中的数据。再以图 1.3 中的经纪公司数据库为例。假如一位用户想要查找签署了某份合约的经纪公司。他首先将相应的合约记录定位于 ENGAGEMENTS 节点，然后再通过 Schedule 集合结构确定哪位客户“拥有”该合约记录。最后，通过 Represent 集合结构确认“拥有”该客户记录的经纪公司。只要通过相应的集合结构，正确地导向及访问，用户就能够解决各种问题。

网状数据库的一个优势是快速的数据访问。同时，相比使用层次数据库，用户可以使用网状数据库创建更为复杂的查询。网状数据库的主要缺点就是，用户必须熟悉数据库的结构，才能通过集合结构来访问。再思考一下图 1.3 中的经纪公司数据库。如果用户要决定某特定合约是否已经付款，那么他就必须熟悉相应的集合结构。网状数据库的另一个缺点就是，很难在不影响与之交互的应用程序的条件下，改变数据结构。记住，在网状结构中，一种关系被明确定义为一个集合结构。你没法做到改变集合结构，却不影响采用这种集合结构查找数据的应用程序。如果你改变一个集合结构，就必须同时改变应用程序中所有对该结构的引用。

尽管与层次数据库相比，网状数据库已经取得了显而易见的进步，但是

数据库界里的少数人却坚信存在更好的管理和维护大量数据的方法。随着新数据模型的出现，用户们提出了更为复杂的问题，继而对数据库有了更高的要求。于是，我们迎来了关系数据库模型（relational database model）。

关系数据库模型

关系数据库的想法最早出现在 1969 年，迄今为止仍然是数据库管理中最为广泛的数据库模型。20 世纪 60 年代，“关系数据库之父”埃德加·F·科德（Edgar F. Codd）博士曾是 IBM 公司的研究员，当时他正在寻找处理大量数据的新方法。对当时的数据库模型和数据库产品的不满，让他开始思索怎样运用数学原理和结构去解决困扰他已久的种种问题。作为一个专业数学家，他坚信自己能够运用数学的特殊分支来解决各种问题，诸如数据冗余、数据完整性差，以及数据库结构过度依赖其物理实现等问题。

1970 年 6 月，科德博士在其题为“大型共享数据库的关系数据模型（A Relational Model of Data for Large Shared Databanks）”¹这一里程碑式的作品中，正式提出了他的新式关系数据模型。他建立的新模型基于两个数学分支：集合论（set theory）和一阶谓词逻辑（first-order predicate logic）。实际上，模型本身的名称取自“关系（relation）”这个术语，它是集合论的一部分。（一个广泛存在的误解就是，关系数据模型是因为关系数据库中的表可以彼此联系而得名。）

关系数据库将数据存储于关系中，用户视之为表（table）。每个关系由元组（或记录）以及属性（或字段）组成。（下文将统一采用术语“表”、“记录”和“字段”。）表中的记录或字段的物理顺序不重要，且表中每个记录由一个具备唯一值的字段进行识别。正是关系数据库的这两种特征使得数据能够不依赖于它在计算机中的物理存储方法而存在。也正因如此，用户不必为了检索数据而了解一个记录的物理位置。这与层次和网状数据库都不同，在

¹ Edgar F. Codd, “A Relational Model of Data for Large Shared Databanks,” *Communications of the ACM*, June 1970, 377 - 87.

层次和网状数据库中，了解结构布局对于检索数据至关重要。

关系模型将关系分为一对一、一对多以及多对多（详见第 10 章“表关系”）。两个表之间的关系是通过匹配一个共享字段的值来隐性建立的。例如，在图 1.5 中，CLIENTS 和 AGENTS 表通过一个 Agent ID 字段产生关系；特定客户通过一个匹配的 Agent ID 与经纪人相关联。同样，ENTERTAINERS 和 ENGAGEMENTS 表通过一个 Entertainer ID 产生联系；ENTERTAINERS 表中一个记录可以通过匹配 Entertainer ID 与 ENGAGEMENTS 表中一个记录相关联。

只要用户熟悉数据库中表之间的关系，访问数据的方式就几近无限。他既可以从直接相关联的表访问数据，也可以从间接相关联的表访问数据。不妨看看图 1.5 所示的经纪公司数据库。尽管 CLIENTS 表与 ENGAGEMENTS 表间接相关联，用户还是可以生成一个客户和对应艺人的列表。（当然，它确实取决于表实际的构建方式，不过这不是我现在要谈的问题。而这个例子能解释我们当下在讨论的问题。）他之所以可以轻而易举地做到这一点，是因为 CLIENTS 直接与 ENGAGEMENTS 联系，而 ENGAGEMENTS 间接与 ENTERTAINERS 联系。

Agents

Agent ID	Agent First Name	Agent Last Name	Date of Hire	Agent Home Phone
100	Mike	Hernandez	05/16/11	553-3992
101	Greg	Johnson	10/15/11	790-3992
102	Katherine	Ehrlich	03/01/12	551-4993

Clients

Client ID	Agent ID	Client First Name	Client Last Name	Client Home Phone
9001	100	Stewart	Jameson	553-3992
9002	101	Susan	Black	790-3992
9003	102	Estela	Rosales	551-4993

图 1.5 关系数据库中相关表的示例

Entertainers

Entertainer ID	Agent ID	Entertainer First Name	Entertainer Last Name
3000	100	John	Slade
3001	101	Mark	Jebavy
3002	102	Teresa	Weiss

Engagements

Client ID	Entertainer ID	Engagement Date	Start Time	Stop Time
9003	3001	04/01/12	1:00 PM	3:30 PM
9009	3000	04/13/12	9:00 PM	1:30 AM
9001	3002	05/02/12	3:00 PM	6:00 PM

图 1.5 关系数据库中相关表的示例 (续)

检索数据

使用结构化查询语言 (SQL)，你就可以在关系数据库中检索数据。SQL 是用于创建、修改、维护以及查询关系数据库的标准语言。如下所示为一个 SQL 查询语句的样例，你可以使用它来生成一个埃尔帕索市 (El Paso) 所有客户的列表：

```
SELECT ClientLastName, ClientFirstName, ClientPhoneNumber
FROM Clients
WHERE City = "El Paso"
ORDER BY ClientLastName, ClientFirstName
```

基本 SQL 查询的三个组成部分是 SELECT...FROM 语句、WHERE 子句，以及 ORDER BY 子句。使用 SELECT 子句指出你想要在查询中使用的字段，FROM 子句则表明该字段所属的表。借助 WHERE 子句对一个或多个字段施加限制，你可以过滤查询返回的记录，然后利用 ORDER BY 子句按照升序或降序将检索结果排序。

如今大多数主流关系数据库软件都包含了各种形式的 SQL 实现，用户不仅可以在窗口中手动输入“原始”SQL 语句，而且各式工具的出现，也让用户可以使用花样繁多的图形元素建立查询。例如，用户使用 R:BASE

Technologies 公司的 R:BASE 时, 可以选择直接在命令提示符中创建和执行 SQL 查询语句, 而 Microsoft SQL Server 用户会发现使用 SQL Server 的图形化查询生成器 (graphical query builder) 建立查询更为容易。不管如何创建查询, 用户都能保存查询, 供日后使用。

使用数据库并不一定必须了解 SQL。如果数据库软件为你提供了图形化查询生成器, 或者你使用定制的应用程序操作数据库中的数据, 就根本不需要编写 SQL 语句。不过, 对 SQL 有基本的认识于你有利无害。这可以让你在使用查询生成工具的时候, 理解所创建的查询, 并为之排错。假如你要利用高端数据库软件, 比如 Oracle 和 Microsoft SQL Server, 了解 SQL 也一定会为你带来好处。

❖注意: 尽管对 SQL 进行详尽讨论已超出本书的范围, 但是你应该明白, SQL 是一种与关系数据库模型直接相关的语言。如果你想要或需要学习 SQL, 不妨先从我的第二本书, 即 *SQL Queries for Mere Mortals®* 的第二版入手, 然后再去阅读本书附录“推荐书目”中列出的与 SQL 相关的其他书籍。

关系数据库的优势

与之前的模型相比, 关系数据库拥有诸多优越之处, 比如以下几点:

- **内置多层次完整性:** 数据完整性 (data integrity) 被构建到模型中, 在字段级确保数据的准确性; 在表层次确保记录不被复制以及检测主键值的缺漏; 在关系层次确保两个表之间的关系有效; 在业务层次确保从业务本身而言数据准确无误。(随着设计过程的展开, 本书将对完整性进行详尽论述。)
- **数据在逻辑和物理上都独立于数据库应用:** 无论是用户更改数据库的逻辑设计, 还是数据库软件供应商更改数据库的物理实现, 都不会对建立在该数据库上的应用程序带来不利影响。
- **确保数据一致性 (data consistency) 和准确性 (accuracy):** 由于施加于数据库各层次上的完整性, 数据能保持一致且精准。(当你完成

- 设计过程时，就会发现这是显而易见的。)
- **简便的数据检索 (data retrieval)**：根据用户的命令，用户可以从数据库中的特定表或者许多相关联的表中检索数据。这使得用户可以以近乎无穷多的方式查看数据信息。

事实证明，以上以及其他的一些优势为商界人士和所有需要收集和管理数据的人们带来了好处。实际上，关系数据库已成为多数情况下的首选。

人们普遍认同关系数据库存在的一个缺点就是，基于关系数据库的软件的运行速度非常缓慢。这并非关系模型本身的毛病，而是由于模型引入的时候辅助技术的问题。处理速度、内存和外存的根本不足，导致没法为数据库软件开发商提供一个能完全实现关系数据库的合适的平台，因而早期的关系数据库软件完全没法发挥出它们应有的潜能。不过，过去 20 年间硬件技术和软件工程两方面取得的突破已经让处理速度变得不足为虑，而开发商们也取得了重大的进展，他们已经能够为这个模型提供更为充分的支持。

当你读完本书中所呈现的设计过程，你就会对关系数据库模型产生更为全面的了解。你将读到的内容包括创建表、确立数据完整性、运用关系以及建立业务规则 (business rule)。

关系数据库管理系统

关系数据库管理系统 (RDBMS) 是一种应用程序，你可以用它来创建、维护、修改以及操作关系数据库。许多 RDBMS 程序也提供创建终端用户应用程序所需的工具，这些应用程序与存储在数据库中的数据进行交互。当然，一个 RDBMS 的质量直接影响到它对关系数据库模型的支持程度。即使在“真正的” RDBMS 中，各个开发商对关系数据库的支持也有所不同，完全地实现关系模型的潜能还需时日。尽管如此，所有 RDBMS 程序仍在不断地完善，而且比以往任何时候都更为强大，功能也更为全面。

在关系数据库发展之初，RDBMS 都是为大型计算机编写的（一切不都是源于大型机吗？）。20 世纪 70 年代初期流行的两个 RDBMS 程序分别是

System R 和 Interactive Graphics Retrieval System (交互图形与检索系统, INGRES), 前者由 IBM 在其加州圣何塞研究实验室开发, 后者由加州大学伯克利分校开发。这两个程序为关系模型受到广泛认可做出了巨大贡献。

随着越来越多的人认识到关系数据库的优点, 许多公司决定由层次和网状数据库模型逐渐向关系数据模型转变, 因此产生了对更多更好的大型计算机 RDBMS 程序的需求。20 世纪 80 年代见证了许多适用于大型计算机的商用 RDBMS 程序的开发, 其中就有甲骨文 (Oracle) 和 IBM 这两家公司开发的商用 RDBMS 程序。

20 世纪 80 年代早期到中期, 个人计算机兴起, 随之而来的是基于 PC 的 RDBMS 程序的开发。早期开发这种产品的公司包括阿斯顿-泰特 (Ashton-Tate) 和 Fox Software 公司, 但是, 它们当时开发出来的程序都只不过是基于文件的数据库管理系统。伴随着 Microrim 和 Ansa Software 公司开发出来的产品, 真正基于 PC 的 RDBMS 程序开始出现。这些公司推动了数据库管理理念和潜力的传播, 使之由大型计算机主导的信息系统领域转向普通终端用户的台式机。

20 世纪 80 年代末至 90 年代初, 随着越来越多的用户使用数据库, 共享数据的需求变得日益显著。由中央数据库支持多个用户, 这个概念似乎是一个很有前景的想法。这必定会使数据管理以及数据库安全性能更容易实现。数据库开发商, 诸如微软和甲骨文, 都通过开发客户端/服务器 RDBMS 程序回应这一需求。

在客户端/服务器环境中, 数据存储计算机中, 该计算机则充当数据库服务器, 用户通过其计算机中的应用程序或者数据库客户端与数据进行交互。数据库开发者则使用客户端/服务器 RDBMS 程序创建并维护数据库以及相应的终端用户应用程序。开发者在数据库服务器上实现数据完整性和数据安全性, 从而在相同数据集的基础上创建各种用户应用, 而不会影响数据完整性或安全性。

后关系模型

尽管 RDBMS 在典型业务应用程序中已被广为接受, 比如库存控制、患者管理、银行业务、订单处理以及事件调度, 但是事实证明它们在计算机辅助设计 (CAD)、地理信息系统 (GIS) 以及多媒体存储系统方面依然有所欠缺。为了应对这一问题, 最终出现了两个新型数据库模型: 面向对象 (object-oriented) 数据库以及对象关系 (object-relational) 数据库。

面向对象的模型包含了面向对象程序设计语言的所有特征, 而且从本质上将关系数据库降低到数据存储状态。这里的基本理念就是数据库开发者操纵数据库的每方面, 包括操作面向对象数据库软件内部数据库中的数据的操作集。数据库软件 and 应用程序软件之间不再有明确的区分。(与其他所有模型一样, 这种方法也是有争议的。) Versant 公司和 IBM 就是两家面向对象数据库软件开发商。

关系模型基于两个不同的数学分支, 具有坚实的理论基础。与此不同, 面向对象数据库模型没有明确的理论基础。同样, 人们对其定义也不具备非凡或有凝聚力的共识。不过, 从某种程度上来说, 由对象管理组织 (OMG) 给出的关于这个模型的范本, 成为了面向对象数据库管理系统的实际标准。

◆注意: OMG 是一个非营利性的国际协会, 专门解决对象标准的问题。它成立于 1989 年, 成员公司超过 800 个。值得一提的是, OMG 并不是像美国国家标准协会 (ANSI) 一样的标准机构, 而只是一个咨询认证团体。

另一方面, 对象关系模型 (object-relational model), 曾叫扩展关系数据模型 (extended relational data model), 通过混合各种面向对象元素和特征, 包括类、封装以及继承, 扩展了关系数据库模型。它的理念是这些扩展将使关系数据库能够管理和操作更为复杂的数据类型, 比如音频流、视频短片和建筑图纸。包括 IBM、甲骨文、微软以及 PostgreSQL 全球开发组在内的开发商们基于这种模型推出了应用程序。

未来将会如何

近几年来，数据库的使用方式得到了极大的发展。随后许多企业都开始意识到，它们可以从存储在各种关系和非关系数据库中的数据处收集到大量有用的信息。这促使它们思考是否存在一种方式——可以进行数据挖掘并获得有用的分析信息，然后，它们能利用这些信息制订重要的业务决策。此外，它们在思考是否可以将数据巩固并整合成企业的可行知识库。的确，这些问题都很难回答。

IBM 曾提出了数据仓库（data warehouse）的构想。按照当初的设想，数据仓库让企业能够访问存储在许多非关系数据库中的数据。他们的首度尝试未能成功，主要原因是与这一任务相关的复杂性和性能问题。直到 20 世纪 90 年代，数据仓库的实现才变得更为实际和可行。比尔·恩门（Bill Inmon）被人们誉为数据仓库之父，他是这项技术的有力推动者和倡导者，对其发展演进起到了重要的作用。现在，随着企业开始利用它们多年以来存储在数据库中的海量数据，数据仓库也变得更为常见。

互联网对公司使用数据库的方式已经产生了显著的影响。许多公司和企业都在使用网络拓展客户群，它们与消费者共享及从消费者收集的很多数据都存储在数据库中。开发者通常使用可扩展标记语言（XML）收集和整合来自各种关系和非关系系统的数据。许多开发商投入大量精力让其客户创建数据库，并把数据存储在“云”中，即一个完全脱离客户机的平台。其构想就是客户可以通过互联网随时随地访问来自“云”数据库的数据。鉴于过去几年里相关设备的广泛出现和使用（相对本书而言），数据库管理系统在这种环境中将如何发展实在值得一看。

最后一点

现在，RDBMS 已经有数十年的历史了，它们继续对人们、企业以及组织与数据的交互方式产生着重大的影响。随着通过互联网访问数据变得越来越容易，而企业又在以前所未有的速度加速其在网络空间的扩展，RDBMS

的作用也会随之不断扩充和发展。无数公司在关系数据库系统上投入了巨资，因此它们不太可能在短期内消失。

小结

本章开篇即定义了当前在数据库管理中使用的两种数据库：操作型数据库和分析型数据库。

然后，对层次数据库模型和网状数据库模型做了简要讨论。讨论内容包括数据结构、关系、分别使用的数据访问方法，以及各自的主要缺点。读者从中能了解到，这两种模型在数据库管理出现初期曾一度得到过广泛应用，并最终促成了关系数据库模型的开发和推广。

接着，我们详细探讨了关系数据库模型，以及其发展历史和特征。我们注意到，它的理论基础是数学的两个分支，正是这个理论基础使得关系模型结构非常稳固。然后，我们探索了这个模型的数据结构和关系，以及 SQL 在模型内部访问数据时所发挥的作用。无疑，读者会记住 SQL 是关系数据库所使用的标准语言。在这一节的结尾，我们回顾了关系数据库模型的优点。

之后，我们简单回顾了关系数据库管理系统的历史，从 20 世纪 70 年代初期的大型计算机系统，到 80 年代的基于 PC 的系统，再到 90 年代的客户端/服务器系统。在这一点上，读者应该意识到时代的进步造就了今日数据库系统的发展。

接下来就是对于对象关系和面向对象数据模型的简要讨论。读者可以发现，这些模型的出现明显是为了应对先进的数据库应用程序，它们都融合了各种面向对象的元素和特征。

最后，我们以数据仓库和通过互联网访问数据的讨论结束本章。读者应该认识到，数据仓库是用于统一和整合来自异构数据源的数据的，而对这些数据的真正利用也是直到最近才变得切实可行。然后，读者应该了解，XML 是收集来自关系和非关系数据源数据的通用工具，而且现在“在云中”存储和管理数据的趋势也愈演愈烈。读者应该意识到，很长一段时间内人们可能

还将继续使用关系数据库，尽管互联网给公司和企业使用数据库的方式带来了巨大的影响。

在下一章中，我们将讨论为什么要关注数据库设计以及为什么理论很重要，还将涉及优秀设计的目标和优点。

思考题

1. 说出当今世界所使用的两种主要数据库类型。
2. 分析型数据库存储的是哪种数据？
3. 判断题：操作型数据库主要用于联机事务处理（OLTP）的情景。
4. 关系数据库模型出现之前，通用的是哪两种数据模型？
5. 描述父/子关系。
6. 什么是集合结构？
7. 说出关系模型所基于的一个数学分支。
8. 关系数据库如何存储数据？
9. 说出关系数据库中的三种关系类型。
10. 如何在关系数据库中检索数据？
11. 指出关系数据库的两个优点。
12. 什么是关系数据库管理系统？
13. 对象关系模型的前提条件是什么？
14. 数据仓库的目的是什么？

第2章 设计目标

第2章 设计目标

从某种意义上来说，一切的事实都为原理。
天空的蓝色呈现出色彩学的基本规律。
寻找现象背后的本质毫无意义——现象即本质。
——歌德（Goethe）

本章内容

为什么要关注数据库设计
理论的重要性
学习优秀设计方法学的益处
优秀设计的目标
优秀设计的好处
数据库设计方法
规范化
小结
思考题

为什么要关注数据库设计

一些使用关系数据库管理系统（RDBMS）应用程序的读者，可能会提出疑问：为什么要关注数据库设计呢？毕竟，大多数程序都有示例数据库，用户可以根据自己的需要来复制和修改这些数据库，甚至还可以从示例数据库中借用表，在自建的其他数据库中使用。另外，一些程序也提供工具引导你定义和创建表。不过，这些工具并不能帮助你设计数据库，它们只是帮你创建数据库包含的物理表。

你必须明白，最好在创建完逻辑数据库结构之后，再使用这些工具。RDBMS 程序提供设计工具和示例数据库，帮助你实现数据库的物理结构所需的时间减到最少。理论上，减少实现时间，则有更多时间创建终端用户程序。

但是，关注数据库设计是因为，数据库设计对于数据库中数据的一致性、完整性和准确性至关重要。如果数据库设计不当，就将难以检索到特定类型的信息，甚至有可能得到错误的检索信息。得到错误的检索信息大概是数据库设计不当产生的最严重后果，因为它能够对你的企业造成实质性的伤害。事实上，如果数据库影响到机构日常业务的开展方式，或将对机构未来的业务开展方向造成影响，就必须关注数据库设计。

我们不妨暂时从另外一个角度来看这个问题：试想一下你会如何着手修建定制住宅。首先会做什么呢？无疑，你不会马上聘请承包商，让他按照他的意愿修建你的住宅。你肯定会先请一位建筑师来设计，然后再请承包商修建。建筑师要先了解你的需求，然后用一系列的图纸将其呈现出来，并记下你定好的各种系统（结构系统、机械系统、电力系统）的规模、形状以及相应的要求。接下来，承包商将雇佣人手并采购包括上述系统在内的材料，然后根据图纸和说明组装好这些东西。

现在，我们再回到数据库这个视角，不妨将逻辑数据库设计看作建筑图纸，而将物理数据库的实现看作完工的住宅。逻辑数据库设计描述的是数据

库的规模、形状以及必要的系统，它解决的是业务信息和运营需求。然后，使用 RDBMS 程序构建逻辑数据库设计的物理实现。一旦创建了表，建立了表关系，并且将数据完整性提升到相应的水平，数据库构建也就完成了。接下来就可以设计和创建应用，让你和你的用户借助数据库中存储的数据轻松实现交互。你可以确信，这些应用将为你提供及时且准确的信息。

尽管你也可以在一个 RDBMS 上随便实现一个数据库设计，但优秀设计远为有利，因为优秀的设计能生成准确的信息，并且可以更加高效地存储数据，还更容易管理和维护。

理论的重要性

❖注意：本章使用术语“理论”代表“作为原理使用的一般命题”，而非“猜想或建议”。

许多重要原理（及相关设计方法学）都具备某种理论基础。结构工程师使用物理学理论设计出无数种不同结构，作曲家使用音乐理论中的概念创作美妙的交响乐和管弦乐作品，汽车产业使用空气动力学理论设计更为高效节能的汽车，而航空航天工业使用这种理论设计轻灵的机翼。

这些例子都表明了理论具有重大意义。理论的主要优点在于帮助人们预测结果；如果你采取某种或一系列的特定行动，理论能帮助你预测接下来会发生的事情。你知道扔一块石头，它必会落地。如果你身手敏捷，是完全可以暂时克服牛顿的万有引力定律向上跳跃的。不过，重点是理论永远都在起作用。如果将一块石头雕刻平整，放在另一块平整的石头上，你可以想见它将待在你放置它的地方。这个理论让你能够设计金字塔、大教堂以及砖石建筑。现在来想想数据库的例子。我们不妨假设有两个关联表。由于关系数据库理论的工作方式，你知道自己可以同时从两个表中提取数据。从两个表中提取数据的依据是两个表共享字段的匹配值。同样，行为产生的结果可预测。

关系数据库基于数学的两个分支——分别是**集合论**和**一阶谓词逻辑**。正因有了理论支撑，关系数据库才能保证信息的准确性。数学的这两个分支，

也构成优秀的设计方法学的基础，以及创建优秀关系数据库结构所必需的基石。

为了完成一个看似相当简单的任务，去研究复杂的数学概念，你大抵是不乐意的，而这也可以理解。另外，你一定还听到过这样的抱怨，说关系数据库及其相关的设计方法学所基于的数学理论与现实世界脱节，或者说这些理论一定程度上就是不切实际的。事实并非如此：数学是关系模型的中心，也保证了模型的可行性。不过，别担心——你并不需要为了使用关系数据库，而去参透集合论或一阶谓词逻辑。这就像你没必要为了开车，而去通晓空气动力学一样。虽然空气动力学理论也许能有助于你理解汽车是如何省油的，但是它可没法帮你学会如何平行泊车。

数学理论为关系数据库模型提供了基础，并使之可预测、可靠且健全。理论描述了创建关系数据库所需的基础建构模块，并为数据库的组织架构提供了指南。所谓“设计”，就是指组织建构模块以达到理想的结果。

学习优秀设计方法学的益处

人们可以通过反复试错，学会如何正确设计数据库。不过，这需要花费很长的时间，在此期间还需要你不断地去纠错。最好的方法就是先学习优秀数据库设计方法学——比如本书中的这种设计方法学，然后再着手设计数据库。

学习和使用优秀设计方法学有以下几个优点。

- 提供设计健全数据库所需的技能。许多数据处理问题的出现都可以归因于冗余数据、重复数据以及无效数据的存在，或者所需的数据缺失。这些问题都会导致产生错误信息，致使查询或报表难以运行。然而，通过采用优秀设计方法学，你基本上可以规避掉所有的这些问题。
- 提供一系列组织有序的技术引导你逐步完成设计过程。这些有序组织的技术确保你能在设计各个方面做出明智的决策。
- 帮助你将失误和设计重复降至最低。无疑，在设计数据库时，你会犯错。但是，优秀的方法学会帮助你识别设计中的错误，并为你提供改

正这些错误的工具。另外，方法学中技术的组织条理性能让你避免无谓地重复给定的设计过程。

- 让设计过程变得更为简单，并减少设计数据库所花时间。无论采取哪种反复试错方式设计数据库，你都难免会浪费宝贵的时间，因为它欠缺优秀方法学所能提供的逻辑性和条理性。
- 帮助你更充分、更高效地理解和使用 RDBMS 应用程序。随着正确设计知识的扩展和提升，你实际上已经开始明白为什么某种 RDBMS 提供特定工具以及如何使用这些工具实现 RDBMS 程序中的结构。

不管你使用本书中所提到的还是其他业已确立的设计方法学，你都应当选择一种设计方法学，并竭尽所能地去学习它，再如实地用它来设计数据库。

优秀设计的目标

为了设计优秀可靠的数据库结构，你必须达到一些明确的目标。如果你在设计数据库时，牢记以下这些目标并一直着眼于实现这些目标，上述章节中所提到的很多问题就都可以避免。

- 数据库应支持设定的和即时的信息检索。数据库必须存储必要的数据，支持在设计过程中满足设定的信息需求，以及满足用户会提出的一切即时查询需求。
- 正确且高效地构建表。数据库中每个表都代表一个主题，由相对独立的字段组成，都应尽可能地减少冗余数据，且都在整个数据库中借助具有唯一值的字段进行识别。
- 数据完整性落实到字段、表以及关系层次。这些层次上的完整性有助于确保数据结构及其值一直准确和有效。
- 数据库支持与组织相关的业务规则。数据必须提供对业务有意义的信息，保证这些信息准确、有效。
- 数据库适应未来发展。随着业务信息需求的变化和发展，数据库结构应易于修改或扩展。

有时，你也许会觉得这些目标难以实现。但是，一旦你实现了这些目标，

你就一定会对自己最终设计出的数据库结构感到满意。

优秀设计的好处

花费在设计健全的数据库结构上的时间肯定是值得的。从长远来看，优秀的设计能为你节省不少时间，因为草率地设计出来的差劲的结构总是故障不断。如果采用优秀的设计技术，你将获得以下好处：

- **数据库结构易于修改和维护。**修改某个字段或表不会对该数据库中其他字段和表造成不利影响。
- **数据易于修改。**修改一个表中给定的字段值不会对该表中其他字段值造成不利影响。此外，因为设计优良的数据库会将重复字段降至最少，所以你通常只需修改某个字段上的一个特定的数据值即可。
- **信息易于检索。**因为表结构构建良好，表之间的关系设置正确，你能轻易地创建查询。
- **终端用户应用易于开发和创建。**你可以将更多的时间花在编程和解决手头的数据操作任务上，而无须为设计拙劣的数据库层出不穷的问题所困扰。

数据库设计方法

传统设计方法

一般而言，数据库设计的传统方法分为三个阶段：需求分析（requirements analysis）、数据建模（data modeling）以及规范化（Normalization）。

需求分析阶段涉及考察待建模企业，进行用户和管理人员调查，以评估当前系统、分析未来需求并评估企业整体的信息需求。这个过程相对直接，确实，本书中所展示的设计过程也遵循相同的思路。

数据建模阶段则涉及使用数据建模方法构建数据库结构，比如实体关系（ER）图、语义对象建模（semantic-object modeling）、对象角色建模（object-role

modeling) 以及 UML 建模。每种建模方法都提供了一种直观展示数据库结构各个方面的方法, 包括表、表关系以及关系特征。事实上, 本书中使用的建模方法是基本 ER 图。图 2.1 所示为一个基本 ER 图的示例。

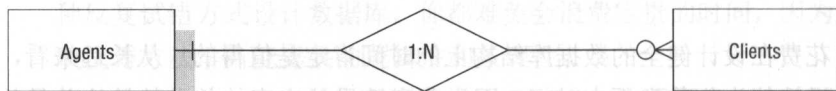


图 2.1 基本 ER 图示例

❖注意: 在本书中, 将数据建模方法融入了设计过程本身, 而不是单独来处理它。接下来, 将酌情介绍并解释整个过程中的每种建模技术。

每种建模方法都包含了一套图表符号, 代表数据库的结构和特征。例如, 图 2.1 中就提供了数据库多个方面的信息。

- 长方形代表两个表, 分别称作 AGENTS 和 CLIENTS。
- 菱形代表这两个表之间的一种关系, 菱形中的“1:N”则指明它是一种一对多关系。
- AGENTS 表旁边的垂直线表示一个客户必须与一位经纪人相关联, CLIENTS 表旁边的圆圈则表示一位经纪人不一定要与客户相关联。

在数据建模阶段, 需要定义字段并将之与相应的表联系起来。每个表都指定一个主键 (primary key), 确认并实现各个层次的数据完整性, 并通过外键 (foreign key) 建立关系。一旦初始表结构完成, 并根据数据模型建立好关系, 就可以进入规范化阶段了。

规范化是将大表分解为小表的过程, 目的在于减少冗余数据和重复数据, 并避免插入、更新以及删除数据时产生问题。在规范化过程中, 表结构对照范式 (normal form) 进行测试, 如果发现任何上述的问题, 就会予以修正。范式就是一系列特定的规则, 可以用来测试表结构, 确保其健全且不会出现问题。现在存在有一定数量的范式, 每个范式都被用于测试一系列特定的问题。当前使用的范式是第一范式、第二范式、第三范式、第四范式、第五范式、第六范式、BC 范式以及域键范式 (Domain/Key Normal Form)。

本书中所呈现的设计方法

本书中所采用的设计方法是笔者过去数年间开发的，它将需求分析和简单 ER 图方法结合起来，以对数据库结构进行图解。然而，它并不包含传统规范化过程，也没有涉及范式的使用。原因很简单：范式会让没有精心学习过正统关系数据库理论的人感到困惑。例如，看看如下第三范式的定义。

关系变量是 3NF，当且仅当它为 2NF 且每个非主属性不传递函数依赖于主键。¹

对于不熟悉关系变量、3NF、2NF、非主属性、传递函数依赖以及主键这些术语的读者，这种描述就毫无意义。

设计数据库的过程并不也不应该让人难以理解。只要清楚明了地把整个设计过程呈现出来，并且解释清楚所有的概念和技术，任何人都应当具备正确设计数据库的能力。例如，以下定义就是从利用第三范式检查表结构的结果中导出的，相信大部分人都会觉得这样的表述直观、易懂。

一个表应该具有一个唯一识别其每条记录的字段，且表中的每个字段应描述该表所表示的主题。

笔者形成这个定义的过程也就是开发整个设计方法学的过程。

规范化

20 世纪 80 年代末，笔者突然意识到关系模型的存在差不多有 20 年了，而人们使用同一个基础方法学设计数据库大约也有 12 年了。（笔者更为惊讶的是，20 多年之后我们还在使用关系模型。）当时，笔者使用的是传统的设计方法学，但是偶尔会觉得不太好用。困扰笔者最多的两件事情就是规范化过程（整体而言），以及实现正确的设计所需完成的、看似无穷无尽的迭代。

¹ C. J. Date, *An Introduction to Database Systems*, 7th ed. (Boston: Addison-Wesley, 2000), 362; emphasis added.

当然，这也是让笔者所认识的大多数其他数据库开发者感到头痛的地方。所以，这绝不是笔者一个人遇到的问题。笔者为此思考了很长时间，然后终于想出了一个解决方案。

笔者意识到，规范化的目的是将设计拙劣或有误的表转化为具有健全结构的表。笔者也清楚其过程：取一个给定表，对照范式对其进行测试，然后判断它是否设计得当。如果设计不当，则加以适当修改，之后再测试，并重复这个过程直到表结构完善。图 2.2 展示了笔者当时对这一过程的设想。

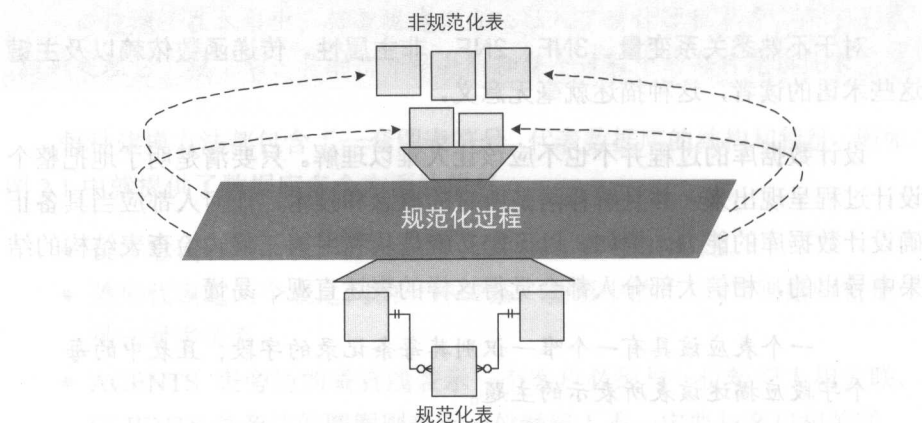


图 2.2 我设想的一般规范化过程的示意图

我将这一切牢记在脑海里，然后提出了以下这些问题。

1. 如果我们假设一个完全规范化的表的设计正确有效，那么，难道我们不能找出这一类表的特征，然后将这些特征定为理想表结构的属性吗？
2. 难道我们不能将这一理想表作为整个数据库设计过程中创建的所有表的模型吗？

毫无疑问，这两个问题的答案都是肯定的，于是笔者开始认真地创建“新”设计方法学的基础。通过找出设计优良的数据库（它成功通过了每个范式的测试）的最终特征，笔者首先为创建健全的结构编译了不同类型的指南。然后，进行了一些测试，同时使用新指南为新数据库创建了表结构，并修复了已有数据库的表结构缺陷。这些测试非常顺利，所以笔者决定将这项

技术应用到整套传统设计方法学当中。于是制定指南，以解决其他与传统设计方法相关的问题，比如域、子类型、关系、数据完整性以及参照完整性。完成新指南之后，笔者进行了更多的测试，发现自己的方法学挺奏效。

笔者的设计方法学的主要优点是，它摆脱了传统设计方法学中许多会让数据库开发新手望而生畏的东西。例如，传统意义上的规范化现在对于开发者而言变得简单明了，因为它（借助新指南）已融入整个设计过程。另一个主要优点是，这种方法学清晰且易于执行。我认为这主要归功于，我用平直的语言表述所有的指南，让大多数人能很轻松地理解这些指南。

你要明白，重要的是，只有当你像在运用其他设计方法学的时候一样，忠实地遵循这种设计方法学，它才能产生完全规范化的数据库结构。缩减、规避、弱化或者省去其中的任何一个部分，你都不可能开发出健全的结构——这也适用于使用任何其他的设计方法学。为了获得预期的回报，必须努力、有条不紊且充分地完成这个过程。

❖注意：本书附录 G，即“关于规范化”中更为详尽地解释了笔者是如何将规范化融入笔者的设计方法学之中的。

还有一些基本术语是在深入研究设计过程之前所必学的，笔者将在下一章中介绍。

小结

本章以探讨数据库设计的重要性开篇。读者应当已经意识到，对于数据库中数据的完整性和一致性而言，数据库设计至关重要。不当或拙劣的设计会造成一个主要问题是**不准确的信息**。恰当的设计至关重要，因为不当的设计会给企业产生错误的信息，造成不利的影响。

接着，我们讨论了理论的重要性及其与关系数据库模型的关联。读者从中应该懂得，模型的数学理论基础赋予了数据库模型十分健全、可靠的结构。

之后，我们考察了学习设计方法学会带来的益处。此外，使用优秀方法

学还会产生高效可靠的数据库结构，减少设计数据库所花的时间，并避免因设计不当所造成的典型问题。

接下来，我们列出了优秀设计的目标。实现这些目标对于成功的数据库设计过程来说至关重要，因为它们有助于确保设计的数据库结构健全。然后，我们又罗列了优秀设计的好处，读者可以发现，花在设计健全的数据库结构上的时间物有所值。

最后，我们以对传统数据库设计方法的简要讨论结束本章，阐释了本书中所呈现的设计方法的前提以及规范化。至此，我们明白了传统设计方法较复杂，需要花费一定的时间去学习和理解。本书中所使用的设计方法简单明了，不仅易于实施，而且所达到的效果与传统设计方法无异。

思考题

1. 使用 RDBMS 程序设计工具的最佳时间是什么时候？
2. 判断题：对于数据的一致性、完整性和准确性而言，设计至为关键。
3. 数据库设计不当最恶劣的后果是什么？
4. 什么使关系数据库结构健全并能够确保信息准确？
5. 说出学习设计方法学的两个好处。
6. 判断题：如果理解了数据库设计，则能更有效地使用 RDBMS 程序。
7. 说出优秀设计的两个目标。
8. 什么有助于确保数据结构及其值一直有效且准确？
9. 说出运用优秀设计技巧的两个好处。
10. 判断题：缩减部分设计过程，仍然能够达成优秀且健全的设计。

第3章 术语

“我要用一个词的时候，”矮座椅沙发用一种相当轻蔑的口吻说，“我想让它是什么意思它就是什么意思——恰如其分。”

——刘易斯·卡罗尔 (Lewis Carroll)

《爱丽丝魔镜之旅》(Through the Looking Glass)

本章内容

术语为何重要

关于值的术语

关于结构的术语

关于关系的术语

关于完整性的术语

小结

思考题

在着手学习设计过程之前，理解本章中的术语非常重要。实际上，还有其他的术语需要学习，本书将会在你的学习过程中提到。书后也有一个术语表，供读者记忆和查阅。

术语为何重要

与其他领域、产业还有学科一样，关系数据库设计也有其特有的术语体系。学习这些术语重要的原因如下。

1. 用于表达和定义关系数据库模型的特殊思想和概念。大部分术语都取自数学的两个分支，即集合论和一阶谓词逻辑，而它们也正是关系数据库模型的基础。
2. 用于表达和定义数据库设计过程本身。一旦理解了这些术语，设计过程就变得容易得多了。
3. 用于讨论关系数据库或 RDBMS。这些术语经常出现在专业杂志、软件使用手册、教学材料、商业数据库软件书籍，以及与数据库相关的网站上。

本章囊括了大部分在设计过程中用以定义思想和概念的术语，包括每个术语是如何定义的，以及对该术语的详细解释。（对于专门运用于设计过程中特定技术的术语，我给出了与之相关的详细解释或必要的深入探讨。）还有一些其他的术语，我将在本书随后的内容里对其进行介绍并讨论，因为我觉得放在与之相关的特定思想或概念的语境中，更便于读者理解。

❖注意：词汇表中包含了本章乃至本书中所有术语的简明定义。

本章中定义的术语分为四类：关于值的术语（value-related）、关于结构的术语（structure-related）、关于关系的术语（relationship-related），以及关于完整性的术语（integrity-related）。

关于值的术语

数据

存储在数据库中的值就是**数据**（data）。数据是静态的，因为它始终保持着相同的状态，除非通过某种手动或自动的方式对其进行修改。图 3.1 展示了一些样本数据。

George Edleman 92883 05/16/96 95.00

图 3.1 基本数据的例子

此时，这个数据并无意义。例如，难以确定“92883”所代表的含义。是邮政编码？还是零件编号？即使它代表一个用户标识号，但是它就是乔治·爱德曼的号码吗？除非等到处理该数据，否则我们无从得知。

信息

信息（information）就是你以这样一种方式进行处理的数据——让其在被使用和观察的时候变得有意义和有效用。相对于存储在数据库中的数据，信息总是在不断地发生变化，另外，你还可以使用无数种方式对其进行处理和表示，所以从这两点来说，它是动态的。你可以将信息显示为 SQL SELECT 语句的运行结果，在计算机屏幕上的表单中显示，或者打印成纸质的报表来显示。需要记住的是，处理数据就必须使之能转化为有意义的信息。

图 3.2 展示了如何将上述例子中的数据处理并转化为有价值的信息。通过这样一种方式处理后——此例中，就是将数据作为病人花费报表的一部分进行处理，这些数据也就有意义了。

理解**数据**和**信息**之间的差异非常重要。设计数据库的目的是给企业和组织内部的人提供有用的信息。只有当数据库中存在相应的**数据**，且该数据库的结构以某种方式支持该**信息**，数据库才能提供这些信息。如果你忘记了数据和信息之别，只要回想起下面这句简单的话就可以了：

存储的是数据，检索的是信息。

当你完全理解了这句话的含义，数据库设计过程背后的原理也就变得通透明白了。

❖注意：不幸的是，在整个数据库产业中，数据和信息这两个术语仍然被频繁地互换使用——这是一种错误的做法。这种错误不仅出现在专业杂志、商用数据库软件书籍和网站上，而且本该精通此道的作者也会有误用的情况。


 Eastside Medical Clinic 7743 Kingman Dr. Seattle, WA 98032 (206) 555-9982			Patient Name: <u>George Edelman</u> Patient ID: 10884 Visit Date: <u>02/16/12</u> Physician: Daniel Chavez		
Doctors Services	Service Code	Fee	Nursing Services	Service Code	Fee
<input checked="" type="checkbox"/> Consultation	<u>92883</u>	119.00	<input type="checkbox"/> R.N. Exam	89327	
<input checked="" type="checkbox"/> EKG	92773	<u>95.00</u>	<input type="checkbox"/> Supplies	82372	
<input type="checkbox"/> Physical	98377		<input type="checkbox"/> Nurse Instruction	88332	
<input type="checkbox"/> Ultrasound	97399		<input type="checkbox"/> Insurance Report	81368	

图 3.2 数据转化为信息的示例

空值 (null)

一个 null 代表一个缺失或未知的值。你必须从一开始就明白，null 并不代表零，也不代表一个包含一个或多个空格的字符串。理由非常简单。

- 零具有多种含义。它可以表示账户结余的状态，当前剩余头等舱机票的数量，以及当前特定产品的库存量。
- 尽管一个包含一个或多个空格的字符串对我们大多数人来说，确实毫无意义，但是它对 SQL 这样的查询语言却意义匪浅。就 SQL 而言，

一个空格是一个有效字符，一个由三个空格组成的字符串（' ' '）就和一个由三个字母组成的字符串（'abc'）一样合法。在图 3.3 中，一个空格代表华盛顿特区并不属于任何县。

- 一个零长度字符串，即两个连续的单引号（''），对于 SQL 之类的语言也是可接受的值，并且在特定情况下也具有意义。例如，在一个 EMPLOYEES（员工）表中，字段中一个被称作 Middle Initial 的零长度字符串值，它也许表示某个员工名字不含有中间名缩写。

Clients

Client ID	Client First Name	Client Last Name	Client City	Client County	State	<< other fields >>
9001	Stewart	Jameson	Seattle	King	WA
9002	Susan	Black	Poulsbo		WA
9003	Estela	Rosales	Fremont	Alameda	CA
9004	Timothy	Ennis	Bellevue	King	WA
9005	Marvin	Russo	Washington		DC
9006	Kira	Bently	Portland		OR

图 3.3 包含 null 的示例表

❖注意：由于版面所限，本书无法总是将所有给定样表的所有字段都展示出来。不过，本书会将与当下正在讨论的问题关系最紧密的字段呈现出来，并且使用“other fields”来表示与示例不甚相关的字段。你会在本书所列的许多示例中，看到这样的表示方法。

Null 的值

当你按照其既定的意义对其加以利用，null 就会非常有用。图 3.3 中的 CLIENTS 表显然就说明了这一点。在出现了 null 的每条记录中，Client County 字段中的每个 null 都代表一个缺失或未知的县名。为了正确地使用 null，首先必须理解它存在的含义。

缺失值（missing value）通常是人为失误造成的结果。比如，以苏珊·布莱克（图 3.3 中 Susan Black）的记录为例。如果你为布莱克女士输入数据，却并未提及其所居住的县名，则在该记录中该数据被视为缺失并以一个 null

来表示。不过，一旦你意识到了这个错误，就可打电话给布莱克女士询问她的住址，然后补充这一信息。

表中出现未知值 (unknown value) 的原因很多。其中一个原因可能是一个字段需要的特定值尚未明确。例如，当前学校课程表数据库中可能有一个 CATEGORIES 表，表中未包含你想在秋季开始开设的一系列新课程。另一个原因是它们的确是未知的。再参照一下图 3.3 中的 CLIENTS 表，以马文·拉索 (Marvin Russo) 的记录为例。比如，你正在输入马文·拉索的相关数据，你问他居住在哪个县，他不知道，而你恰巧也不知道他住的市到底属于哪个县，那么其记录中的县字段所对应的值就是未知的，在该记录中用一个 null 表示。显然，只要你们两人中有一人能确认正确的县名，就可以解决这个问题。

如果一个字段的所有值都不适用于某特定的记录，则该字段的值也可能为 null。假定某个时候使用一个 EMPLOYEES 表，其中包含一个 Salary (薪水) 字段和一个 Hourly Rate (每小时工资) 字段。这两栏中有一栏的值始终为 null，因为一个员工不可能同时拥有固定薪水和按小时计算的工资。

值得一提的是，“不可用”和“不适用”之间有非常细微的区别。在刚才的例子中，两个字段中有一个字段的值实际上不可用。假定现在有一个 PATIENTS 表，其中包含一个 Hair Color (发色) 的字段。你更新这个表中当前一位男性患者的记录。如果该患者最近没有头发了，那么该字段所对应的值就是“不适用”。尽管只要用 null 就能代表一个不适用的值，但是笔者始终还是推荐使用诸如“N/A”或“Not Applicable (不适用)”的真实值来表示。从长远来看，这让信息表达得更为明确。

正如你所看到的，表中是否能出现 null 取决于你使用该数据的方式。了解了使用 null 的好处后，我们不妨再来看看使用它会带来的不利影响。

Null 所带来的问题

Null 的主要缺点在于对数学运算有不利影响。包含 null 的运算所得的值也为 null。这中间的逻辑合理清楚——如果一个数未知，则该运算的结果必

然也未知。注意如下的例子中，null 是如何改变运算结果的：

$$(25 \times 3) + 4 = 79$$

$$(null \times 3) + 4 = null$$

$$(25 \times null) + 4 = null$$

$$(25 \times 3) + null = null$$

图 3.4 中 PRODUCTS（产品）表有助于解释 null 对数学表达式的影响，这些数学表达式使用了表中的字段。其中，Total Value 字段所对应的值得自数学表达式 “[SRP] × [Qty On Hand]”。观察这个表中的记录时，你会注意到在 Qty On Hand 的值为 null 的地方，所对应的 Total Value 字段的值也会缺失，亦即 Total Value 字段的值也为 null。这导致一个严重的漏检错误，即当 Total Value 字段中所有值相加时，会出现一个不准确的总和。这个错误之所以“漏检”，是因为 RDBMS 程序本身不会提醒你这个错误。唯一能够避免这个错误的办法就是确保 Qty On Hand 字段的值不为 null。

Products

Product ID	Product Description	Category	SRP	Qty On Hand	Total Value
70001	Shur-Lok U-Lock	Accessories	75.00		
70002	SpeedRite Cyclecomputer		65.00	20	1,300.00
70003	SteelHead Microshell Helmet	Accessories	36.00	33	1,118.00
70004	SureStop 133-MB Brakes	Components	23.50	16	376.00
70005	Diablo ATM Mountain Bike	Bikes	1,200.00		
70006	UltraVision Helmet Mount Mirrors		7.45	10	74.50

图 3.4 上表中的 null 会影响到涉及表中字段的数学运算

图 3.5 则有助于我们理解 null 对包含表中给定字段的值的统计函数的影响。如果一个统计函数基于包含 null 的字段，则该函数的结果为 null，比如 Count(<字段名>)。图 3.5 中的表展示了一次汇总查询的结果，计算的是图 3.4 中 PRODUCTS 表的每个类别的出现总次数。Total Occurrences（出现总次数）字段的值是函数表达式 Count([Category])的计算结果。注意：如果汇总查询显示未指定类别的出现次数为“0”，则表示每种产品都已指定一个类别。但

是，这个信息明显不准确，因为 PRODUCTS 表中还有两种产品未指定类别。

在数据库设计过程中，缺失值、未知值以及某个值是否用于数学表达式或统计函数中，都应当纳入考虑范围，我们将在后面的章节中再回顾并讨论这些问题。

Category Summary

Category	Total Occurrences
	0
Accessories	2
Bikes	1
Components	1

图 3.5 null 对统计函数的结果有影响

关于结构的术语

表

根据关系模型，我们知道关系数据库中的数据都存储在关系中，用户视之为表。每种关系由元组（记录）和属性（字段）组成。图 3.6 展示了一个典型的表结构。

Clients

Client ID	Client First Name	Client Last Name	Client City	<< other fields >>
9001	Stewart	Jameson	Seattle
9002	Susan	Black	Poulsbo
9003	Estela	Rosales	Tacoma
9004	Timothy	Ennis	Seattle
9005	Marvin	Russo	Bellingham
9006	Kira	Bently	Tacoma

记录

字段

图 3.6 一个典型的表结构

表是数据库的主要结构，每个表始终代表一个单独的特定主题。表中记录和表中字段的逻辑顺序根本不重要，每个表包含至少一个称为主键的字段，给予每条记录独一无二的标记。（例如，在图 3.6 中 Client ID 为 CLIENTS 表的主键。）实际上，关系数据库中数据的存在之所以能独立于它们在计算机中的物理存储方式，正是因为表最后这两个特征。对于用户而言，这是特大好消息，因为用户无须知道记录的物理位置就可以检索其数据。

一个给定表代表的主题可以是一个对象或一个事件。当主题为一个对象时，就意味着该表表示某种有形的东西，比如一个人、地点或者事物。抛开其类型不谈，每个物体都有其独特的特征可以作为数据存储下来，然后以近乎无穷多种方式将这些数据作为信息来进行处理。飞行员、产品、机器、学生、建筑物，以及装备都是表可以表示的物体。图 3.6 展示了一个最为常见的这种类型的表。

如果表的主题是一个事件，则意味着该表表示发生在指定时间点的某个事件，它具有你想要记录的特征。你可以将这些特征存储为数据，然后采用与上述情况完全相同的方式，将之作为信息来处理。需要记录的事件可以包括司法听证、资金分配、实验室测试结果，以及地质调查等等。图 3.7 展示了一个表示事件的表的示例。我们都或多或少地经历过这个事件——预约医生。

Patient Visit

Patient ID	Visit Date	Visit Time	Physician	Blood Pressure	<< other fields >>
92001	02/14/12	10:30	Hernandez	120/80
97002	02/14/12	13:00	Black	112/74
99014	02/15/12	09:30	Rolson	120/80
96105	02/15/12	11:00	Hernandez	160/90
96203	02/15/12	14:00	Hernandez	110/75
98003	02/16/12	09:30	Rolson	120/80

图 3.7 表示事件的表

一个存储数据以提供信息的表称为**数据表 (data table)**。数据表是关系数据库中最为常见的一种表。这种表中的数据呈动态变化，因为数据可以操作（修改、删除等等）也可以采用某种形式或方式处理成信息。人们使用数据库时，总是在不断与这些类型的表进行交互。

另一方面，**验证表 (validation table)**，也称为**查找表 (lookup table)**）存储专门用于实现数据完整性的数据。验证表通常用来表示主题，例如城市名称、技巧类别、产品编码以及项目识别号码等。这种表中的数据是静态的，因为它几乎不发生变化。尽管用户极少直接与这些表打交道，但经常要间接使用它们来验证输入数据表中的值。图 3.8 展示了验证表的一个示例。

Categories

Category ID	Category Name
10000	Accessories
20000	Bikes
30000	Clothing
40000	Components

图 3.8 验证表的一个示例

第 11 章“业务规则”将对验证表做更为详细的讨论。

字段

字段（在关系数据库理论中被称为**属性**）是数据库中最小的结构，它代表表所属表的主题的一个特征。字段是实际存储数据的结构。字段中的数据，可以以你能想到的任何形式来检索和作为信息呈现出来。从数据中得到的信息的质量，与确保结构完整性和字段本身的数据完整性所投入的时间成正比。因此，决不能低估字段的重要性。

在设计得当的数据库中，每个字段有且仅有一个值，字段名称会标识它所拥有的值的类型。因此向字段中输入数据非常直观。如果看到诸如 FirstName、LastName、City、State 和 ZipCode 等名称的字段，就能确切知道每个字段中输入的值的类型。你也会发觉，很容易就可以按状态查找数据，

或找到所有姓氏 (last name) 为 “Hernandez (赫尔南德斯)” 的人。

通常，你会在设计不当或差劲的数据库中遇到另外三种类型的字段：

1. 复合字段 (multipart field)，也被称为合成字段 (composite field)，其值中包括两个或两个以上不同的项。
2. 多值字段 (multivalued field)，其包含多个相同类型的值。
3. 计算字段 (calculated field)，包含一个串联文本值或一个数学表达式的结果。

图 3.9 所示表分别列举了每种字段的例子。



Client ID	Client First Name	Client Last Name	Client Full Name	Address	Client City, State, Zip	Account Rep
9001	Stewart	Jameson	Stewart Jameson	Seattle, WA 98125	John, Sandi
9002	Susan	Black	Susan Black	Poulsbo, WA 98370	Frits
9003	Estela	Rosales	Estela Rosales	Bellevue, WA 98005	John
9004	Timothy	Ennis	Timothy Ennis	Seattle, WA 98115	Frits, Sandi
9005	Marvin	Russo	Marvin Russo	Bellingham, WA 98225	Frits, John
9006	Kira	Bently	Kira Bently	Olympia, WA 98504	Sandi

图 3.9 上表中包含了常规字段、计算字段、复合字段和多值字段

在第 7 章 “建立表结构” 中，我将对计算字段、复合字段以及多值字段做更为细致的介绍。

记录

在表中，一个记录（在关系数据库理论中也被称为元组）代表某个表的主题的一个唯一实例。它包含表中的所有字段，而不管这些字段是否有值。根据定义表的方式，数据库中每条记录都通过其主键字段所对应的唯一值识别。

在图 3.9 中，表中每条记录表示一个唯一的客户，Client ID 字段可以识

别整个数据库中的一个给定客户。反过来，表中的每条记录包含了所有字段，而每个字段则描述该记录所代表的客户的某种特性。比如，看看蒂莫西·恩尼斯（Timothy Ennis）的记录。他的记录是表主题（“客户”）的一个唯一实例，包含了表中所有的字段，可以当成一个单元。这些字段的值则表示关于恩尼斯先生的相关信息，这些信息对于组织中的某个人来说会非常重要。

记录是理解表关系的关键因素，因为人们需要了解一个表中的某个记录如何与其他表中的记录产生联系。

视图

视图（view）是一个虚表，由数据库中一个或多个表的字段组成，组成视图的表称为**基表**。在关系模型中，视图之所以被称为虚表，是因为它只是从基表中获取数据，而不存储数据。事实上，视图存储在数据库中的唯一信息就是其结构。许多主要的 RDBMS 程序支持视图，但是其中一些（比如 Microsoft Access）将之视为已保存的查询。至于读者将之视为视图还是已保存的查询，将视具体使用的 RDBMS 程序而定。

视图能让你从多个不同的方面查看数据库中的信息，为使用数据带来了很大的灵活性。你可以用各种各样的方式来创建视图，特别是当这些视图是建立在多个相关表的基础上时，它们会非常有用。例如，在一个学校安排表的数据库中，你可以创建一个视图，用以合并 STUDENTS（学生）、CLASSES（课程）以及 CLASS SCHEDULES（课程安排表）表中的数据。

图 3.10 包含一个名为 INSTRUMENT ASSIGNMENTS（器械分配）的视图，其中包含了取自 STUDENTS、INSTRUMENTS（器械）以及 STUDENT INSTRUMENTS（学生器械）表中的字段。这个视图展示了同时从所有表中所提取的数据，这是基于 STUDENTS 和 STUDENT INSTRUMENTS 表中 Student ID 字段，以及 INSTRUMENTS 和 STUDENT INSTRUMENTS 表中 Instrument ID 字段的匹配值之上的。

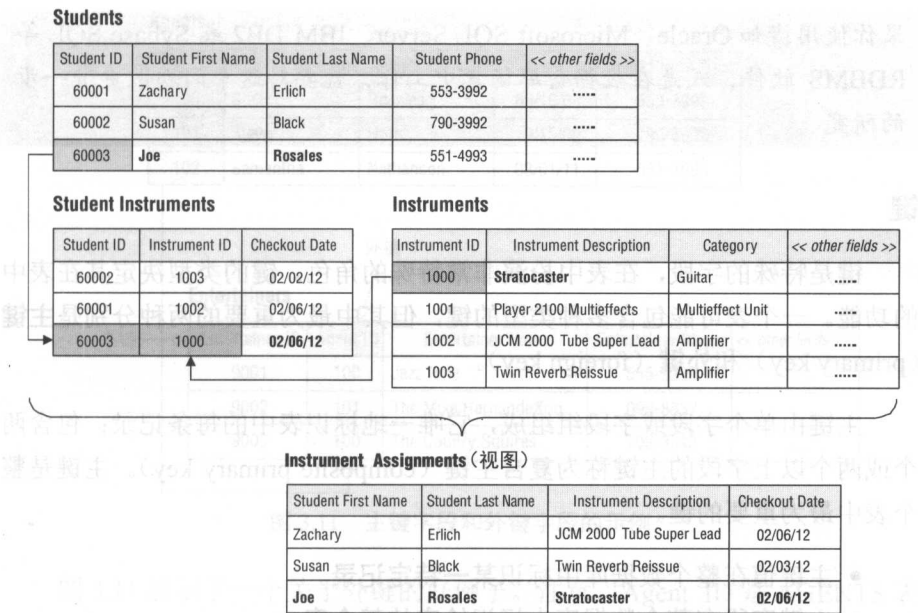


图 3.10 一个典型的视图的示例

视图的重要有以下三方面原因：

1. 可用于同时处理多个表中的数据。（为此，这些表需要彼此关联，或有关系。
2. 可用以防止某些用户查看或操作单个表或一组表中的特定字段。就安全性来说，这一功能非常有用。
3. 可用于实现数据完整性。相应的视图则称验证视图（validation view）。

在第 12 章中，将介绍更多关于设计和使用视图的知识。

❖注意：尽管所有大型数据库供应商都支持本节中所介绍的这种视图，但是现在有一些供应商支持的是索引（或物化）视图。索引视图不同于常规视图之处在于，其本身存储数据。另外，为了提升 RDBMS 处理视图数据的速度，你可以索引其字段。鉴于本书的论述范围，本书将不对索引视图进行全面的讨论，因为它是一个特定供应商的实现问题。不过，如

果你使用诸如 Oracle、Microsoft SQL Server、IBM DB2 或 Sybase SQL 等 RDBMS 软件，或是在数据仓库场景中工作，就要对这个问题做更进一步的研究。

键

键是特殊的字段，在表中扮演非常特殊的角色。键的类型决定其在表中的功能。一个表可能包含多种类型的键，但其中最为重要的两种分别是主键（primary key）和外键（foreign key）。

主键由单个字段或字段组组成，它唯一地标识表中的每条记录；包含两个或两个以上字段的主键称为复合主键（composite primary key）。主键是整个表中最为重要的键。

- 主键值在整个数据库中标识某一特定记录。
- 主键字段在整个数据库中标识给定的某个表。
- 主键实现表层次的完整性，与数据库中其他表建立关系。（在下一节中，你将了解到更多关于关系的知识。）

数据库中每个表都须有一个主键！

图 3.11 中 Agent ID 字段就是主键的一个范例。它能够唯一地标识 AGENTS 表中每个经纪人，并通过确保不出现重复记录，来保证表层次的完整性。它也能够与 AGENTS 表与数据库中其他表之间建立关系，如示例中所展示的 ENTERTAINERS 表。

当你确定两个表之间存在联系时，要建立两者之间的关系，通常需要复制第一个表中的主键，放在第二个表的结构中作为外键，这样才能建立起关系。取名“外键”是由于第二个表已经有了自己的主键，而从第一个表中引入的主键对第二个表来说是“外来物”。

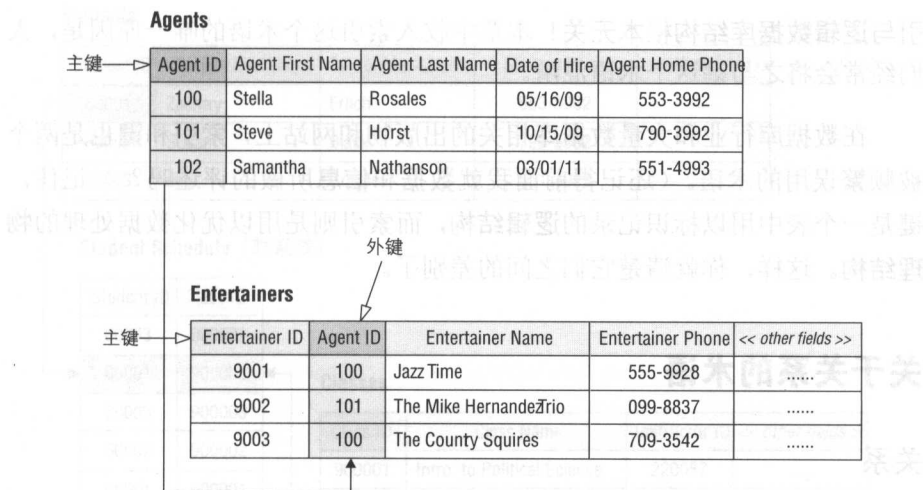


图 3.11 主键字段和外键字段的示例

图 3.11 展示了一个关于外键的好例子。注意，Agent ID 是 AGENTS 表的主键，也是 ENTERTAINERS 表的外键。Agent ID 扮演外键角色，是因为 ENTERTAINERS 表已有主键 Entertainer ID。这样，Agent ID 就建立起了这两个表之间的关系。

除了有助于建立表之间的关系外，外键还有利于实现和确保关系层次的完整性。这意味着两个表中的记录都将始终保持适当相关，因为外键的值必须与其所引用的现有主键值相匹配。关系层次的完整性也有助于避免糟糕的“孤立”记录。孤立记录的一个典型例子就是订单记录找不到对应的顾客。如果不清楚是谁下的单，就不能处理订单，显然也就没法开发票。这会毁掉你一个季度的销售额！

键字段在关系数据库中发挥着重要作用，因此务必学会如何创建和使用键字段。在第 8 章“键”和第 10 章“表关系”中，你将了解到更多关于主键的知识。

索引

索引 (Index) 是由 RDBMS 提供的、用于改善数据处理的一种结构。至于索引如何运作以及如何使用索引则由特定的 RDBMS 程序决定。不过，索

引与逻辑数据库结构根本无关！本章中收入索引这个术语的唯一原因是，人们经常会将之与键这个术语混淆。

在数据库行业和大量数据库相关的出版物和网站上，索引和键也是两个被频繁误用的术语。（还记得前面我就数据和信息所做的评述吗？）记住，键是一个表中用以标识记录的逻辑结构，而索引则是用以优化数据处理的物理结构。这样，你就清楚它们之间的差别了。

关于关系的术语

关系

当使用某种方式关联一表的记录与另一表的记录产生联系时，两表之间就存在一种关系。通过使用一系列主键和外键（按上一节所述方法）或者借助第三个表，即联系表（linking table）（也叫作关联表〔associative table〕），就能建立起表之间的关系。建立关系的方式确实取决于表之间存在的关系类型。（你马上就能了解到更多关于这方面的知识。）图 3.11 展示了通过主键/外键建立起来的一种关系，图 3.12 所示则是借助联系表所建立的关系。

关系是关系数据库的重要组成部分。

- 关系让你能够创建多表视图。
- 对于数据完整性来说，关系非常关键，因为它有助于减少冗余数据和消除重复数据。

每种关系的特征都能通过三种方式描述：表之间存在的关系类型、每个表参与的方式，以及每个表的参与度。

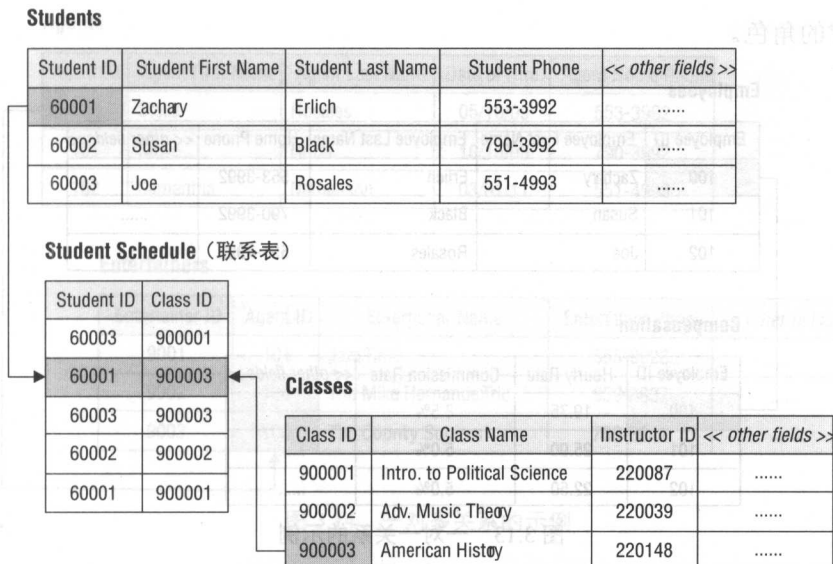


图 3.12 借助联系表建立的两表之间的关系

关系类型

两表之间可存在三种关系（传统叫法为基数〔cardinality〕）：一对一、一对多及多对多。

一对一关系

如果第一个表中的单个记录仅关联至第二个表中的一个记录且反之亦然，则这两表之间就存在一对一关系。在这种关系中，一表作为“父”表，另一表作为“子”表。复制父表的主键，并入子表作为外键，就建立起了这种关系。这是一种特殊的关系，因为它是两表可能共享同一主键的唯一一种关系。

图 3.13 所示为一种典型的一对一关系。其中，EMPLOYEES 表为父表，COMPENSATION(报酬)表为子表。两表之间的关系是这样的，EMPLOYEES 表中的一个记录仅能关联到 COMPENSATION 表中的一个记录上，反之亦然。注意，Employee ID 事实上是两个表的主键。不过，它在子表中也扮演

外键的角色。

Employees

Employee ID	Employee First Name	Employee Last Name	Home Phone	<< other fields >>
100	Zachary	Erllich	553-3992
101	Susan	Black	790-3992
102	Joe	Rosales	551-4993

Compensation

Employee ID	Hourly Rate	Commission Rate	<< other fields >>
100	19.75	3.5%
101	25.00	5.0%
102	22.50	5.0%

图 3.13 一对一关系的示例

一对多关系

如果第一个表中的某一记录与第二个表中的多个记录相关联，但是后一表中的单个记录只与第一个表中的唯一一个记录相关，就把这两表之间的关系称为一对多关系。（描述一对一关系时所用的父/子模型在此处也适用。在本例中，处于一对多关系中所指的“一”端的表为父表，处于“多”端的表为子表。）这种一对多关系的建立，是通过复制父表的主键，写入子表作为外键来完成的。

图 3.14 所示为典型的一对多关系。AGENTS 表中某一记录可与 ENTERTAINERS 表中一个或多个记录相关联，但是 ENTERTAINERS 表中单个记录只能与 AGENTS 表中唯一一个记录相关联。读者可能已经猜到了，Agent ID 是 ENTERTAINERS 表的一个外键。

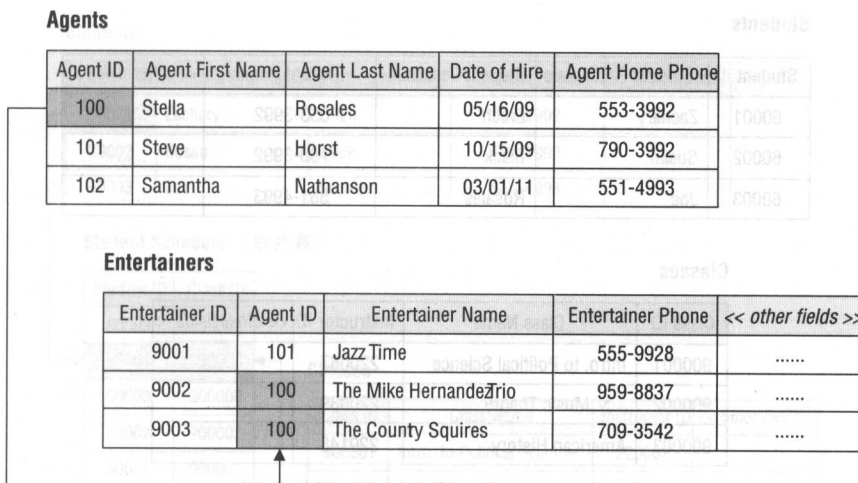


图 3.14 一对多关系的示例

在数据库中，一对多关系目前是两表之间最为常见的一种关系。从数据完整性的角度来说，一对多关系至关重要，因为它有助于消除重复数据，把冗余数据保持在最低的水平。

多对多关系

如果第一个表中的单个记录可以与第二个表中的多个记录相关联，且第二个表中的单个记录可以与第一个表中的多个记录相关联，这两表之间的关系就被称为多对多关系。使用联系表可以建立起这种关系。（本节开头部分对这种类型的表进行过简单的介绍。）使用联系表能轻易地将一表的记录与另一表的记录联系起来，并且能确保在添加、删除或修改相关数据时不会出现问题。通过复制关系中的两表的主键，并用这些字段形成新表的结构，就定义了联系表。这些字段实际上有两个作用：放在一起就构成了联系表的复合主键；一旦分开，它们就各自成为一个外键。

一个未完善建立的多对多联系是“不完整的”。图 3.15 所示为一个典型的不完整的多对多关系。其中，STUDENTS 表中的单个记录可以与 CLASSES 表中的多个记录相关联，而 CLASSES 表中的单个记录也可以与 STUDENTS 表中的多个记录相关联。

Students

Student ID	Student First Name	Student Last Name	Student Phone	<< other fields >>
60001	Zachary	Erlich	553-3992
60002	Susan	Black	790-3992
60003	Joe	Rosales	551-4993

Classes

Class ID	Class Name	Instructor ID	<< other fields >>
900001	Intro. to Political Science	220087
900002	Adv. Music Theory	220039
900003	American History	220148

图 3.15 不完整的多对多关系的示例

这种关系不完整是由多对多关系的固有特性造成的。关键在于：如何轻易地将第一个表中的记录与第二个表中的记录联系起来？根据图 3.15 所示的表来重构这个问题，就变成了如何将单个学生与多门课程或者将特定课程与多名学生相联系。将若干 Student 字段插入 CLASSES 表中？还是将多个 Class 字段添加到 STUDENTS 表中？事实上，这两种方法都会让你难以使用表中的数据，还会对数据完整性造成不利的影响。最好的方法就是创建和使用一个联系表，这样就能以最恰当和最有效的方式建立起多对多关系。图 3.16 所示就是这种方法的实际运用。

重要的是，你必须清楚两表之间存在的关系类型，因为关系类型决定了表与表之间如何联系起来，表之间的记录是否相互依赖，以及可以在关系中存在的相关联记录的最小和最大数量。第 10 章“表关系”中将介绍更多关于关系的知识。

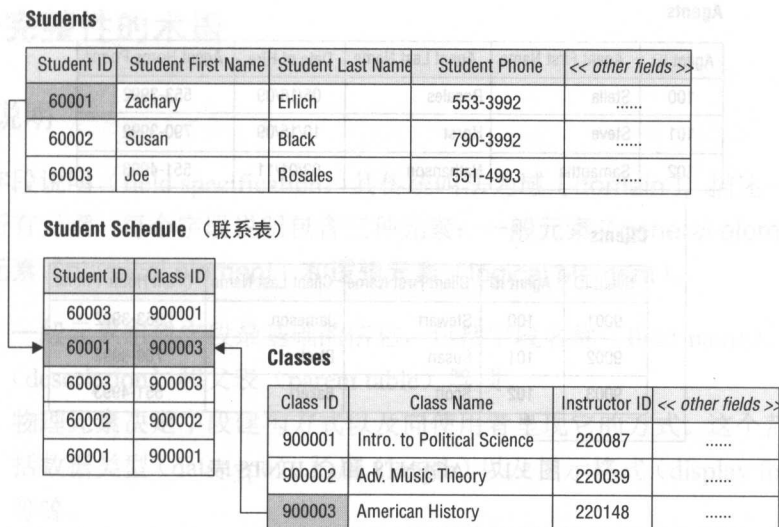


图 3.16 使用联系表建立多对多关系

参与的类型

在一种关系中，表的参与可分为强制的（mandatory）和可选的（optional）。假设 TABLE_A 和 TABLE_B 这两个表之间存在一种关系。

- 如果在向 TABLE_B 输入记录之前，必须向 TABLE_A 输入至少一条记录，那么 TABLE_A 的参与就被视为强制的。
- 如果在向 TABLE_B 输入记录之前，无须向 TABLE_A 输入任何记录，那么 TABLE_A 的参与就是可选的。

如图 3.17 所示，以 AGENTS 表和 CLIENTS 表为例。如果在向 CLIENTS 表中输入一位新客户前，必须存在一位经纪人，那么，该关系中 AGENTS 表的参与就是强制的。然而，如果在向 CLIENTS 表中输入一位新客户前，对于是否存在经纪人并无要求，那么，该关系中 AGENTS 表的参与就是可选的。通过确定与 CLIENTS 表中数据相关的 AGENTS 表中数据的使用方式，就能正确判定 AGENTS 表的参与类型。例如，如果想要确保每位客户都被指派给一个空闲的经纪人，就要让关系中 AGENTS 表的参与变为强制的。

Agents

Agent ID	Agent First Name	Agent Last Name	Date of Hire	Agent Home Phone
100	Stella	Rosales	05/16/09	553-3992
101	Steve	Horst	10/15/09	790-3992
102	Samantha	Nathanson	03/01/11	551-4993

Clients

Client ID	Agent ID	Client First Name	Client Last Name	Client Home Phone
9001	100	Stewart	Jameson	553-3992
9002	101	Susan	Black	790-3992
9003	102	Scott	Baker	551-4993

图 3.17 AGENTS 和 CLIENTS 表

参与度

参与度有两个作用，一是确定某一特定表中必须与关联表中的单一记录相关联的最少记录条数，二是确定某一特定表中被允许与关联表中单一记录相关联的最多记录条数。

再来考虑一下 TABLE_A 和 TABLE_B 之间存在的关系。通过找出 TABLE_B 可以与 TABLE_A 中单个记录相关联的最多和最少记录条数，确定 TABLE_B 的参与度。如果 TABLE_A 中单个记录可以与 TABLE_B 中相关联的记录不小于 1 且不大于 10 个，则 TABLE_B 的参与度为 **1,10**。（参与度的表示方法为，最小数量在左，最大数量在右，中间用逗号隔开。）同理，可以确定 TABLE_A 的参与度。通过找出每个表中数据被关联和被使用的方式，也可以确定某一关系中每个表的参与度。

再回到图 3.17 中的 AGENTS 和 CLIENTS 表。如果要求一个经纪人至少负责一位且至多负责八位客户，则 CLIENTS 表的参与度为 **1,8**。如果要确保一位客户只指派给一个经纪人，那么就将 AGENTS 表的参与度定为了 **1,1**。在第 10 章中，读者将学会如何给某一特定关系设定参与度。

关于完整性的术语

字段说明

字段说明 (field specification, 其传统叫法为域 [domain]) 描述一个字段的所有元素。每个字段说明包含三种元素: 一般元素 (general element)、物理元素 (physical element) 和逻辑元素 (logical element)。

- 一般元素组成字段最基础的信息, 包括字段名称 (field name)、描述 (description) 和父表 (parent table) 等项。
- 物理元素决定字段建构方式以及向使用者呈现它的方式。这个范畴包括数据类型 (data type)、长度 (length) 以及显示格式 (display format) 等等。
- 逻辑元素描述存储在字段中的值, 它包括所需值 (required value)、值的范围 (range of values) 以及默认值 (default value) 等等。

在第9章“字段说明”中, 读者将了解到所有与字段说明相关的元素, 包括这里所提到的内容。

数据完整性

数据完整性 (data integrity) 指的是数据库中数据的有效性、一致性和准确性。毫不夸张地说, 从数据库中检索到的信息的准确度与数据库的数据完整性程度成正相关, 数据完整性是数据库设计过程中最为重要的一个方面, 绝不能低估抑或忽视它, 甚至于连半点疏忽都是容不得的。否则, 就会被难以检测或识别的错误所困扰。这会造成严重的后果, 导致做出重要决策时, 能拿到不准确的信息为依据已是万幸, 最糟糕的情况下, 信息完全无效。

在数据库设计过程中, 要实现四种类型的数据完整性。其中有三种是基于数据库结构的不同方面, 并根据它们起作用的范围 (层次) 来命名的。第四种数据完整性则基于机构认知和使用数据的方式。下面简要介绍这四种数据完整性:

1. 表层次完整性 (table-level integrity, 传统上称作实体完整性) 确保表

中无重复记录，并且确保表中字段对每个记录的标识是唯一的且不是空值。

2. **字段级完整性**（field-level integrity，传统上称作**域完整性**）确保每个字段的结构健全；确保每个字段的值有效、一致且准确；确保相同类型的字段（比如，City 字段）在整个数据库中定义一致。
3. **关系层次完整性**（relationship-level integrity，传统上称作**参照完整性**）确保两表之间的关系是健全的，且无论哪个表中输入、更新或删除数据，两表中的数据始终保持同步。
4. **业务规则**基于企业认知和使用数据的方式，对数据库的特定方面施加限制或约束。这些限制能够影响数据库设计的多个方面，比如字段中存储的值的范围和类型，关系中每个表参与的类型和程度，以及特定关系中用于关系层次完整性的同步类型。所有这些限制都将在第 11 章中进行更为详尽的探讨。因为业务规则能影响完整性，所以在设计过程中，必须和其他三种类型的数据完整性一并考虑。

小结

本章开篇即说明了术语对于定义、讨论以及解读关系数据库模型和数据库设计过程的重要性。

关于值的术语一节展示了数据与信息之间的显著差别，并指出了理解这种差别对于理解数据库设计过程的至关重要性。现在，读者对 null 及其如何影响从数据库中检索的信息应该有了充分的认识了。

接下来的一节中则讲述了**关于结构的术语**，读者从中应该了解了每个关系数据库的核心结构是**字段、记录和表**。视图是虚表，可用于同时处理两个或两个以上表中的数据。然后谈到**键字段**，它用于唯一地识别表中的记录和建立两表之间的关系。最后，阐述了**键字段和索引**之间的区别。至此，读者清楚了从严格意义上来说，索引是用于优化数据处理的一种软件设备。

在关于关系的术语一节中，我们讨论了两表之间的关系。关系有助于确保各方面的数据完整性，它也是视图用来从多个表中提取数据的机制。接着，又提到了表关系的三种特征：**关系的类型**（一对一、一对多、多对多），**参与的类型**（可选的和强制的）以及**参与度**（相关记录的最小数量/最大数量）。

本章以探讨关于完整性的术语结尾。**字段说明**描述了字段的一般特征、物理特征和逻辑特征（在这里，特征指的是数据库中每个字段的必要组成部分）。随后谈到，**数据完整性**是数据库设计过程中最为重要的一个方面，因为它对数据库中数据有显著的影响。此外，还学习了数据库完整性的四种类型——其中三种基于数据库结构，还有一种基于企业解读和使用数据的方式。这些层次的完整性确保了数据库设计的质量，以及从中检索到的信息的准确性。

思考题

1. 术语为什么重要？
2. 说出术语的四种类型。
3. 数据和信息之间有何差别？
4. Null 代表什么？
5. Null 的主要缺点是什么？
6. 数据库的主要结构有什么？
7. 说出三种类型的表。
8. 视图是什么？
9. 指出**键**和**索引**之间的区别。
10. 两表之间可以存在哪三种关系？
11. 哪三种方式可以描述关系的特征？

12. 字段说明是什么?

13. 字段说明包括了哪三种元素?

14. 数据完整性是什么?

15. 说出数据完整性的四种类型。

第 4 章

第 2 部分

设计过程

内容章本

12. 字段说明是什么?

13. 字段说明包括了哪三种元素?



第 4 章

数据库设计

概念性概述

数据库设计

我不会假装自己了解宇宙——因为与之相比，我不过沧海一粟。

——托马斯·卡莱尔 (Thomas Carlyle)

本章内容

完成设计过程的重要性

明确宗旨和任务目标

分析现有数据库

创建数据结构

确定和建立表关系

确定和定义业务规则

确定和定义视图

审查数据完整性

小结

思考题

了解如何设计关系数据库并不像了解宇宙那么困难；实际上要远为轻松。但是，重要的是从整体上认识数据库设计过程的实现方式，以及设计过程中涉及的步骤。本章的目的就是概述数据库设计过程。

为了完成这个概述，本书将设计过程涉及的所有技术整合为7个阶段，并概述了每个阶段。讨论将展现数据库设计过程的总体情况，希望读者能借此更清晰地理解本书第2部分将呈现的所有数据库设计技巧。

读者可以运用本书中的设计方法学，设计全新的数据库，完善现有数据库，或者分析现有数据库，并基于这些分析结果设计一个新的数据库。

❖注意：单人或多人组成的设计团队都可以设计数据库。在下文中，将使用数据库开发者或开发者代指设计数据库的个人或团队。

完成设计过程的重要性

从一开始，我就想着重强调完成设计过程的重要性。过去，我经常被人问及，到底有没有必要完成整个设计过程。对此，我总是高声回答“有必要！”接着人们又会问，如果只是创建一个“简单的”数据库，还有这个必要吗？（简单是数据库开发者最为忌讳的一个词，因为没什么事情是“简单的”。）同样，我的回答还是肯定的——仍然有必要。无论设计的数据库是哪种类型、规模多大以及目的是什么，都不能忽视完整的设计过程的重要性。读者必须从头到尾一步不落地遵循和落实数据库设计过程。

众所周知，试图在未采用全面的数据库设计过程的情况下设计数据库是一个糟糕透顶的想法。许多数据库的问题都是由拙劣的数据库设计造成的，只遵循部分设计过程，与完全不使用设计过程一样糟糕。不完整的设计就是拙劣的设计。唯有贯彻执行完整的设计过程，才能保证数据库结构健全，数据完整。

记住这一点，数据库中结构完整性和数据完整性的水平与贯彻设计过程的程度成正比。花费在设计过程中的时间越少，数据库出现问题的风险也就越大。尽管彻底遵循数据库设计过程不能保证在设计数据库的时候不出现问题，但是这能极大地减少问题产生的概率。使用RDBMS软件的时候，会发现设计优良的数据库更容易实现。

设计数据库并不难，只是设计出规范数据库需要一些时间。所以，即便设计过程太过漫长，也千万不要妄想走捷径。要保持耐心，记住一位先哲曾说过的话：

务求一次成功，否则贻误再三。

明确宗旨和任务目标

数据库设计过程的第一阶段就是明确宗旨（mission statement）和任务目标（mission objective）。宗旨表明了数据库的目标，并为设计工作提供明晰的焦点。

创建数据库都有着特定的目标，有的是为了解决某个业务问题，有的是为了管理组织或企业的日常交易，还有的是作为信息系统的一个组成部分。确定数据库的用途，在宗旨中对其进行定义，将有助于确保开发出合格的数据库结构，以及收集到必要的数据库，从而达到预期的目标。

在这个阶段，也需明确任务目标。任务目标是用户可以对数据库中的数据执行的常规任务。任务目标可以支撑宗旨，有助于确定数据库结构的各个方面。

明确宗旨和任务目标分别涉及两部分人。第一部分人包括数据库开发者（你）、机构所有者或负责人，以及管理人员，他们的责任是明确宗旨。第二部分人包括数据库开发者（还是你）、管理人员，以及终端用户，他们的责任是明确任务目标。

分析现有数据库

数据库设计过程的第二阶段涉及分析现有数据库——当然，是指在现有数据库存在的情况下。组织机构的情况各不相同，现有数据库通常为遗留数据库或纸质数据库。遗留数据库（也被称为继承数据库）已存在并使用了多年。如你所知，纸质数据库是表格、索引卡、统计文件夹以及类似资料的松

散集合。无论数据库的类型或状况如何,加以分析就会产生有价值的信息,了解机构当前使用和管理数据的方式。这种分析也包括审核机构当前收集和展示数据的方式。观察机构如何使用纸笔(通过表格)收集以及(通过报表)展示数据。如果机构使用某种软件应用程序管理和操作数据库中的数据,就要研究其收集和在屏幕上展示数据的方式。最后,还要考虑机构如何在网上使用其数据(前提是存在这一回事),并审核使用该数据库的所有基于浏览器的应用。

分析的另一方面包括开展对用户和管理人员的访谈,了解他们在日常工作中如何使用数据库。作为数据库开发者,你需要了解用户使用数据库的情况及他们当前的信息需求。然后,与管理人员交流,询问他们当前了解到的信息,以及他们对于机构的整体信息需求的认识。这些访谈是分析的重要组成部分,因为从中了解到的(或未了解到的)信息对数据库的最终结构有重大影响。如果想设计一个完全满足该组织机构的信息需求的数据库,就必须开展全面完善的访谈。

接下来,使用从分析和访谈中收集到的信息,编辑一个初始字段列表。然后,移出所有的计算字段并放进单独的列表中,从而精简初始字段列表。计算字段将在后面的设计过程中用到。由此所得的初始字段列表就是该机构的基本数据需求,也是新数据库设计的起点。(如你所知,任何事情都没有绝对终点。随着设计开发的深入,将继续扩展和改善这个字段列表。)

初始字段列表完成后,立刻发送到用户和管理人员手中,让他们进行简单的审核并提出可能的修改意见。鼓励他们反馈信息,考虑他们的修改建议。如果建议合理有据,则进行适当修改,记录下当前状态下的列表。然后进入下一阶段。

创建数据结构

为数据库创建数据结构是数据库设计过程的第三阶段。包括定义表和字段,建立键,以及为每个字段定义字段说明。

表是数据库中定义的第一个结构。首先，由第一阶段确立的目标以及第二阶段收集的数据需求，来确定表将表示的各个主题。然后，为这些主题建立表，并将它们与第二阶段编辑的字段列表中的字段相匹配。完成这项任务之后，再审核每个表，确保每个表只表示一个主题且不包含重复字段。

现在，继续审核每个表中的字段。提炼表中所有的复合字段和多值字段，确保它们分别只包含单一值，并将与该表所表示的主题特征不符的字段移出或删除。审核完成之后，接着审核和改进表的结构。这涉及审核之前的字段工作，目的是避免遗漏任何细节，确保每个表的结构定义合理。然后，为每个表建立合适的键。任务的重心是确保每个表都拥有正确定义的主键；这个特殊的键唯一标识表中的每个记录。

这个阶段的最后一步是为每个字段建立字段说明。在此要开展用户和管理人员访谈，了解他们看重的具体字段特征，审核和讨论他们不熟悉的特征。访谈完成后，就可以为每个字段定义和记录字段说明。之后，与用户、管理人员再次审核表的结构和字段说明，并适当做出改进。一旦完成审核中发现的所有需改善之处（如果存在），表结构即可进入下一阶段。

确定和建立表关系

第四阶段就是建立表关系。包括再次开展用户和管理人员访谈，发现关系，确定关系特征，以及建立关系层次完整性。

与用户和管理人员接触需要认真谨慎，他们能够帮助你发现数据中的关系。你无法做到对数据的方方面面都非常熟悉，所以参考他们对数据的认识对你非常有利。

确定关系之后，就要使用主键或联系表建立每种关系中表之间的逻辑联系。具体由表之间关系的类型决定。接下来，确定每种关系中表的参与类型和参与度。在某些情况下，由于表中存储数据的性质，这些参与特征会很明显。在其他情况下，依据特定业务规则可以确定参与特征。

确定和定义业务规则

确定和定义业务规则是数据库设计过程的第五阶段。在这个阶段中，将要开展访谈，确定数据库各方面的限制，建立业务规则，以及定义和实现验证表。

机构认知和使用数据的方式，将决定一系列写入数据库的限制和要求。与用户和管理人员的访谈能帮助你确定施加在数据、数据结构和关系上的限制。然后，将这些规范确立为业务规则并记录。

对用户进行访谈，能够揭示数据库各个方面的具体限制。例如，某用户使用订单处理数据库，他对具体细节非常了解。比如，发货日期一定比订单日期晚；必须有日间电话号码；始终要标明运送方式，等等。另一方面，与管理人員的访谈则透露出数据库各个方面的一般限制。例如，某娱乐机构的业务经理熟悉许多业务常识，比如一个经纪人最多能接手 20 个艺人，每年必须更新每位艺人的宣传资料。

下一步，定义和实现必要的验证表，以支持某些业务规则。假如机构以特定的方式使用某些字段，则会发现其值在某范围内。在这种情况下，就可以使用验证表，确保这些字段存储的值一致且有效。

业务规则在此时建立的完整性层次就显得意义重大了，因为它直接关系到机构认知和使用数据的方式。随着机构发展壮大，对数据的看法将发生变化，这就意味着业务规则也会随之改变。确定和建立业务规则是一个持续、重复的过程，要想适当维持这个层次的完整性，就必须不断努力。

确定和定义视图

设计过程的第六阶段是确定和定义视图。在这个阶段，将再次开展访谈，确定使用数据库的方式，以及建立视图。

通过与用户和管理人員交流，了解他们如何使用各自的数据，就能确定

数据库中需要建立的视图类型。例如，你可能发现很多用户开展工作需要详细的信息，而其他人则只需要概要信息帮助他们制定机构的战略决策。每个用户群体都必须以各自不同的方式访问信息，开发者可以使用视图来解决这些问题。

接下来，就是使用适当的表和字段，定义访谈过程中确定下来的视图，并根据特定信息的检索要求为这些视图建立标准。比如，为某个视图建立标准，要求就是必须列出所有位于得克萨斯州的顾客或者必须显示华盛顿州每个城市授权供应商的总数。

审核数据完整性

第七也是最后一个阶段就是从数据完整性出发审核数据库的最终结构。

首先，审核每个表，确保其符合正确设计的标准；检查每个表中的字段，确保结构合理。然后，解决遇到的问题和不一致之处，并再次审核其结构。适当改进完成之后，就可以检查表层次完整性。

其次，审核和检查每个字段的字段说明。先对字段进行必要改进，再检查字段级完整性。这次审核是为了再次确认之前确定和建立的字段级完整性。

再次，检查每种关系的有效性，确认关系类型，以及确定关系中每个表的参与特征。然后，审核关系层次完整性，确保共享字段之间存在匹配值，以及关系中两表之间不存在插入、更新或删除数据的问题。

最后，审核之前确定的业务规则并确认数据库各方面的限制。如果最后一次访谈过后，发现了新的限制，就可以将之确立为业务规则，添加到已有的业务规则中。

完成整个数据库设计过程后，就可以在 RDBMS 程序中实现数据库结构。然而，这个过程并未真正达成圆满，因为随着机构的发展，数据库结构始终需要改进和完善。

小结

本章开头即讨论了完整数据库过程的重要性。读者应该从中认识到，设计数据库时，缺乏优良的设计方法，就会导致设计不规范。同时，还讨论了结构完整性和数据完整性的层次与设计过程的规范程度成正比。数据不一致和信息不准确，是设计不规范的数据库时常常会出现的两个问题。

接下来，我们概述了整个数据库设计过程。为了清晰地展示设计数据库的各个步骤，该过程整合为以下几个阶段。

1. **明确宗旨和任务目标。**宗旨表明了数据库的目标，任务目标则阐明了用户可用数据库中数据执行的任务。
2. **分析现有数据库。**通过审核机构当前收集和展示数据的方式，了解机构数据要求；通过对用户和管理人员进行访谈，了解他们日常使用数据库的方式。
3. **创建数据结构。**确定数据库的主题，再创建表。字段表示对应表的主体特征。将字段与对应的每个表联系起来，并设计一个特定的字段（或一组字段）作为主键。接着为表中的每个字段建立字段说明。
4. **确定和建立表关系。**确定数据库中表之间存在的关系，并使用主键和外键的组合或联系表为每种关系建立逻辑联系。然后再为每种关系设置合适的特征。
5. **确定和定义业务规则。**开展用户和管理人员访谈，确定数据库中数据的限制。机构认知和使用数据的方式通常决定了数据库限制的类型。然后，将这些限制表述为业务规则，它们将用于建立各个层次的数据完整性。
6. **确定和建立视图。**通过用户和管理人员访谈，确定他们使用数据库中数据的各种方式。完成访谈后，酌情建立视图。使用适当的表和字段定义每个视图，而对于必须展现有限记录集合的那些视图，则

为之建立标准。

7. 检查数据完整性。这个阶段包括四个步骤。第一，检查每个表，确保其满足规范的设计标准。第二，检查所有字段说明。第三，测试每种关系的有效性。第四，检查和确认业务规则。

思考题

1. 为什么遵循完整的设计过程十分重要？
2. 判断题：结构完整性的层次与设计过程的规范程度成正比。
3. 宗旨的目标是什么？
4. 任务目标是什么？
5. 读者所属机构的基本数据需求由什么构成？
6. 如何确定各个表所代表的主题？
7. 判断题：为每个字段建立字段说明是在数据库设计过程的第二阶段。
8. 如何为一种关系中的各表建立逻辑联系？
9. 一系列写入数据库的限制和要求由什么决定？
10. 设计和实现什么可以支持特定业务规则？
11. 如何确定写入数据库的视图类型？
12. 在何种情况下，才能在 RDBMS 程序中执行逻辑结构？

第 5 章

大幕开启

他问道，“我该从哪儿开始呢？陛下。”

“从开始的地方开始吧，一直读到末尾，然后停止。”国王
郑重地说。

——刘易斯·卡罗尔

《爱丽丝漫游奇境记》

本章内容

开展访谈

案例分析：迈克自行车行

明确宗旨

明确任务目标

小结

思考题

万事皆有头，数据库设计过程亦是如此。有趣的是，这个过程开始就是明确最终结果。数据库设计过程的第一步，就是确定和表明数据库的目标。同时，还要定义和表明用户在数据库中的数据上可以执行的任务清单。这两项为开发数据库指明了重点和方向，也有助于确保最终数据库结构支持所描述的目标和任务。

开展访谈

访谈是数据库设计必不可少的一部分。在设计过程的一些阶段，它发挥着关键作用。假定你供职于某机构，需要设计一个数据库来支持你和同事开展工作。你应该确保使用本书中所介绍的方式开展访谈。这意味着，在整个设计过程中要接触到一些同事、管理人员以及企业主（由机构的规模决定）。如果你为一家小机构工作，员工人数不多，或者只是为自己个人创建一个数据库，就要开展“自我访谈”；你仍然是在按照本书所述的方法开展访谈，但是要同时扮演采访者和受访者两个角色。你要自问自答。

❖注意：访谈是一项技巧，需要有一定的耐心，付出一定的汗水并经过一定的实践才能学会。可以借助各种途径和技巧开展访谈，在这方面也有许多学术论文、文章和书籍可供参考。由于这个主题与本书内容无关，故不在此深入讨论。不过，本章中纳入了若干项技巧和指导，有助于高效地开展访谈。

访谈之所以重要，是因为它为开发者和数据库使用对象提供了宝贵的沟通桥梁，能有效确保设计工作最终成功，以及提供能够影响数据库结构设计的重要信息。例如，你在使用表关系时，可能会发现难以确定特定关系的参与类型和参与度。唯一的方法就是在机构内部对适当的人员进行访谈。然后，你可以使用访谈中收集到的信息设置关系特征，将访谈当作信息收集工具，从部分数据库建设参与者处获得新的认识或者弄清楚你不理解的事实。注意，这个设计过程包含的每次访谈都必不可少，无论数据库类型如何以及涉及的人数多少。忽视或省略任意一次访谈，都必将遗漏一些重要信息，从而对数据库的最终结构造成不利影响。

❖注意：在余下的章节中，所有访谈都采用开放式问题，这些问题也是相应概念或技术的一部分。读者可以将之当作指南，在访谈中指导自己如何制定问题。

在开展访谈之前，必须制定指南。这有助于确保访谈连贯流畅，并且始

终（通常）获得成功。以下是读者可以为参与者或自己制定的一些方针。

参与者指南

- **让参与者知晓你的意图。**许多人都对访谈保持警惕。他们不希望“将自己置于聚光灯下”，也不愿意被问到“难言之隐”。对每个人透露你想要讨论的话题、其他参与者的名字、你想要开始对话的时间，以及该访谈是否为持续系列访谈的一部分。如果人们了解开展访谈的方式以及你期待的话题，就都会参与到对话当中，并且认真负责地回答问题。最重要的是让受访者放心，访谈不是变相评估他们的表现，确保他们无拘无束地与你坦诚交谈。需要走很长的路才能与参与者建立起信任基础。
- **让参与者知道你感谢他们的参与，他们对访谈问题的回答对整个设计过程都具有重要价值。**此前的经验可能会使一些人认为，无论在工作中付出怎样的努力，都不会得到注意和赏识。即使他们对某项事业确实做出了特殊的贡献，得到的也不过是句“谢谢”而已。鉴于此，他们确实没动力参与访谈。很多参与者（即便不是全部）开始都持有这样的态度，但如果采访者能够真挚诚恳地感激受访者参与采访，并对受访者的回应展现出十足的兴趣，就能实实在在地提高受访者的积极性。向受访者保证，其反馈对整个过程都具有价值，并且在许多情况下，他们的回答能验证设计过程中做出的决策。坦诚相待，博得参与者的信任，这样他们才更有可能尽力帮助你；你的工作会因此轻松不少，大家都会积极热情地予以配合。在再次访谈中，展现如何运用参与者之前的贡献，也非常有效。
- **如果产生争议，确保每个人都知道你是正式仲裁人。**在访谈中，微小争议总是难以避免。在这些争议得到解决之前，也会出现一定的紧张局面。如果有人对这些争议进行仲裁，就能避免这种情况。数据库开发者是最佳人选，因为你能从客观角度辩证地看待问题。此外，开发者所做出的决策符合数据库结构的最大利益。牢牢记住，对与数据库结构无关的争议，如果有更合适的权威人士，则应咨询权威人士。

采访者指南

- 访谈室应选取光线充足、远离噪音的房间，并配备大桌子和舒适的座椅。注重访谈氛围将极大地提升访谈成功的可能性。这方面的反差将足以对访谈的气氛和参与者的精神面貌带来显著的改变。采用光线充足的房间，因为这样参与者就能够轻松地阅读访谈材料。大桌子确保每个人都有足够的工作空间，舒适的座椅则让大家身体放松，精神集中于访谈。

自笔者编写本书的初版和随后的第二版以来，商业环境发生了巨大的变化。现在，许多人都通过计算机开展远程访谈和会议，地点也更多地选择了公众场所，比如餐厅或者当地的星巴克；如果找不到合适的访谈地点，可以选择这些场所。许多机构更倾向于将特定的会议安排在酒店的会议室，因为他们发现这样能让人们远离自己日常工作的环境。

- 每次访谈限制人数不超过 10 人。限制参与者的人数能营造更为轻松的气氛，也更易于鼓励每个人都参与进来。每次访谈人数太多就会出现一个问题：一些参与者的危机感与参与人数成正比上升。一些人害怕在同事面前显得无知无能——不管存在这种感觉是否合理。因此，应当限制每次访谈的人数。
- 对用户和管理人员分别访谈。将两部分人分开访谈有诸多原因，包括上一项中提到的“恐惧因素”。主要是因为不同人群对机构整体和机构日常使用数据的方式看法不同。在整个数据库设计过程中，对不同人群分别访谈能将他们独特的看法转为优势。另一个原因是，当不同人群在特定问题上产生分歧时，可以避免冲突。两组人之间缺乏沟通是很正常的，访谈很有可能会让这个问题暴露出来，其几率高达 50%。放在一起访谈可能驱使他们建立更好的沟通渠道，但也有可能会导致这个问题进一步恶化。无论如何，这个沟通问题都会使访谈复杂化，影响结果。充分运用对机构的了解可以帮助判断是否需要分开访谈。如果需要同时对两组人群展开访谈，那就制订相应的计划，明确目的，并且为干扰做好准备。
- 如必须对多组人员进行访谈，可为每个组安排一个组长。组长有助于

确保访谈顺利开展。组长将负责为所在组每个成员做准备工作，并提供访谈之外获取到的信息。在访谈期间，组长可以指定准备充分的组员回答问题。

也许，偶尔会遇到某个组长想主导访谈，回答每一个问题。发生这种情况时，委婉礼貌地告诉他，从所有参与者处收集反馈信息的工作（责任）在你身上，这样才能对机构整体信息要求做出全面的评估。如果这么做解决不了问题，可以选择在以后的访谈中排除此人或者指定其他人担任组长。

- **在访谈之前准备好问题。**准备好一系列问题，访谈会变得相当容易。（临场发挥，急中生智并非明智之举，即使采访者经验丰富，非常擅长即兴发挥也不好。）预备好问题能为访谈定下重点和方向，也保证了参与者思维的连贯性。话题在提问中轻松转换之间，访谈也变得更加顺利、更加富有成效。

访谈问题务必采用开放式问题。例如，“你觉得我们的服务(a)差、(b)一般，还是(c)优秀呢？”这是一个封闭式问题。封闭式问题作用不大，因为它只为受访者提供了已有的答案选项，这样他们就不能发表客观意见或详细回答。而开放式问题则远为有效，比如“你对我们的产品印象如何？”，因为受访者可以不受限制地回答这个问题。有时候也许需要采取封闭式问题，不过最好谨慎对待，为之制订相应计划，明确目的。

- **如果你不擅长记笔记，就为每次访谈安排可靠的记录者或经小组许可后使用数字记录器（录音机）对访谈进行记录。**访谈是为相应机构收集特定的信息，所以详细的记录非常重要。如果觉得开展访谈的同时难以做记录，就应该选择一位参与者作为助手，让他做记录。（这也是鼓励通常沉默寡言的人参与进来的一种好方法。）助手须谨慎选择，因为如果他在访谈期间走神，就会毁掉这场记录。另一种选择就是使用录音机进行记录。这也许更为有效，因为录音机能精确捕捉访谈内容，借此能够准确判断指定信息由谁提供。（如果决定采用录音机记录，首先必须获得每个参与者的许可。否则，可能侵犯他人隐私和机密。）

● **给予每个人同等的关注。**必须牢记这一点，全神关注发言者，真诚以待。如果你给参与者留下的印象是无聊、不耐烦或心不在焉，他的投入程度就会随之减少。相反，如果他看到你对他的言论充满兴趣并且十分关注，就会满怀热情地参与进来。

有时，某位参与者可能会做出模糊或不完整的回答。这种表现可能有多种原因。一是，他不知道如何表述自己的观点或者不能透露特定细节。二是，他不愿意谈论自己和自己的行为，或者出于某种原因，比如他对你心存疑虑。

因此，必须保持耐心，让参与者感觉轻松。这样，他就会提供你想要的信息。例如，可以尝试猜测他的意思，让他判断虚实。

● **保持访谈的节奏。**访谈期间，可能会反复讨论某个点或者花费大量时间从相关参与者身上提取信息。通过为每个问题设置个人回答时间以及特定话题设置规定时间，就可以避免这种情况的发生。但是，不要将这个时间限制告诉参与者；而是暗示暂时将之搁置，让访谈继续进行下去。在访谈之后，必须与数据库所有者取得联系，这样就可以针对该问题得出结论和解决方案。

● **始终控制好访谈。**这是访谈指导中最重要的一条。如果丧失对访谈的控制，将不可避免地出现问题。比如，某位参与者谈论与议程主题无关的话题，由此变换访谈的重点。这样就必须立即重新为讨论定调，否则访谈一定会不受控制。在一些情况下，可以轻易夺回对访谈的控制权。如果出现意外情况，就必须声明你的访谈部分已经“完成”，让参与者继续他们的讨论。不过，只要保持对访谈的控制，就不会发生这些事情。

访谈是设计过程必不可少的一部分。在接下来的几章中，笔者将提供一些访谈的实例。使用对话范例展示典型访谈情景和访谈中可能用到的问题示例。（其中的问题示例始终围绕当前访谈展开。）

❖ **注意：**访谈范例的目的在于展示开展特定类型访谈所要使用的技巧。因此，笔者将对话设置得相对简单。在访谈中使用对话作为提出好想法的手段。

最后一点：记住本书中所介绍的指南只是**建议**。读者不可能在每次访谈中都用到所有这些指南，或者按照文中所描述的那样运用。只有在理想的条件下，它们才能完全适用。而读者面对的不可能始终都是理想的环境。不过，还是可以将满足这些要求作为目标。最终将获益良多。

案例分析：迈克自行车行

本书中大量的示例都展示了数据库设计过程中所使用的概念和技术。这些示例取自多个数据库，并被巧妙应用到文中。这种用法主要是为了表明学会如何使用特定概念或技术的**原理**，以应用到正在设计的数据库中。因此，注意力应始终放在所展示的概念或技术上，而不是示例本身。

书中将单个数据库示例当作案例进行分析，展示设计过程中包含的步骤，目的是连贯地展现该过程。随着数据库设计过程的展开，将会使用各项技术为案例中的虚构机构设计数据库。本章只对该机构做了简单介绍，但随着新概念或技术的出现，将会提供更多相关信息。

案例分析的对象是迈克自行车行。这是一家新开的自行车行，位于距西雅图市区不远的郊区绿湖。开业至今只有两个月，业务正稳步增长。车行店主迈克将他的日常业务记录在纸上。他将销售情况记录在预印表格上，员工和供应商信息则记在纸上（放在文件夹中），而老顾客的信息书写在索引卡上。结果，迈克花费了大量时间维持这些数据。他有一台电脑，不过主要用来玩游戏，在 YouTube 上看视频，发电子邮件，在 Facebook 上联系朋友，以及浏览各种高尔夫网站。在电脑上执行的唯一与业务有关的任务是，使用电子制表程序记录车行存货。

最近，迈克认识到数据库是存储和使用与业务相关数据的一个好方法。使用数据库将大幅减少维护数据所花费的时间，也能始终保证数据及时更新和信息准确。尽管他认为这个想法不错，但是他对设计数据库一窍不通。不过迈克毫不气馁，决定请一位数据库技术顾问来为自己设计数据库。

在这个故事中，读者就是他聘请的技术顾问。在接下来的章节中，随着数据库设计过程的展开，读者将运用所学技术为迈克自行车行设计数据库。

学到新概念或技术的同时，迈克会为你提供完成数据库设计所需的信息。

明确宗旨

在前面的章节中，已经提到宗旨用于概括地表述数据库的特定目标，在数据库设计之初就要予以明确。此外，宗旨为设计工作指明了焦点，避免出现偏差，以致数据库结构过大或过于复杂。

优良的宗旨

优良的宗旨简明扼要。表述过于烦琐会让人迷惑，产生误解；这就掩盖了数据库的目标。下面列举一个典型的范例。

新星达人（New Starz Talent）经纪公司数据库的目标是维护数据，以及向客户和艺人分别提供支持预订服务和管理服务的信息。

这条宗旨立意明确条理清晰，毫不拖泥带水。一般的宗旨表述就应当如此。把宗旨当成黑暗隧道尽头的一点烛火。只要找准方向，蜡烛的光芒就会引导你不断深入。同样，宗旨也会引导你到达数据库设计的终点。在宗旨的引导之下，你可以专注于设计出合适的数据库结构，达到数据库所宣称的目标。

优良的宗旨避免直接表述具体任务的词句。如果宗旨中包含此类词句，则应删除或修改宗旨。另外记录下被删除的表述，因为也许会在制订目的的时候用到它们。（下一节将论述如何制订宗旨。）下列是一个表述混乱的示例。

霍特科姆郡听证审查官数据库的目标是记录土地使用申请，维护申请人的数据，记录所有听证会，记录所有决定，记录所有诉求，维护部门员工数据，以及维护一般办公支出数据。

显然，上述宗旨有多处错误。

- 稍嫌啰嗦。记住规范的宗旨表述应简明切要。
- 数据库具体目标不明确。这种表述让人难以确定数据库的具体目标。

- **具体任务过多。**这种结构会造成两个问题。第一，任务描述与明确数据库具体目标无关。第二，这种表述显得不完整。让人不禁要问，“是不是还遗漏了什么任务呢？”

删除具体的任务表述（但请保存于他处以备下一步使用），改写宗旨，就解决了这些问题。下面展示一种改法：

霍特科姆郡听证审查官数据库的目标是维护听证审查办公室使用的数据库，审查官办公室使用这些数据就霍特科姆郡公民提交的土地使用请求做出决策。

注意，其中数据库的目标变得十分明确，而且表述简要，无遗漏之处。设计数据库的过程中，使用这种方式制订宗旨，就要始终坚持明确的中心点。

制订宗旨

制订宗旨的过程包括对企业主或管理者访谈，了解企业，以及确定数据库的目标。

第一步是与企业主或者由其指定的员工访谈。不管访谈对象是谁，都能帮助你明确宗旨，因为他们都对企业有整体认识，总体上立刻理解数据库的必要性。此外，这个访谈还会为企业本身提供大量信息。这些信息富有价值，在后面的设计过程中将会用到。

鼓励受访者讨论企业的方方面面，即使讨论涉及与数据库没有直接关系的问题也没关系。这主要是让你了解企业的职能和运作方式；对企业的了解越深入，准备工作就越充分，设计出来的数据库也更符合要求。一旦更好地了解企业本身，就清楚了企业对数据库的一般要求，从而将这些要求转化到宗旨中。

确保访谈采取开放式问题。在某些情况下，好的问题能在不费力气的情況下，促使参与者说出数据库的目标。例如，提出下列这个问题：

“你如何向新客户描述企业的目标呢？”

这个开放式的问题提得很好，因为它既切中肯綮，又给予参与者充足的自由来组织完整的答案。此外，这种问题通常能引发参与者的回应，而回应能直接转化为宗旨。

假定你得到如下回应：

“我们为客户提供任何场合下的娱乐服务。我们关注所有合约的细节，尽可能使客户无忧。”

这种回答轻易就能改写成对宗旨的表述。如果这种回答包括两个或两个以上句子，则其中一句通常指明了数据库的目标。例如，可以使用上面回应的第一句来制订宗旨。下列是其中的一种：

全明星达人数据库的目标是维护我们使用的数据库，支持面向客户的娱乐服务。

记住，最重要的一点是你（数据库开发者）和数据库使用对象理解宗旨表达的意思。不同人群表述的方法不同，措辞很大程度上由特定产业的术语决定。使用一句话表述数据库的特定目标，并得到相关人员的理解和认同，宗旨也就完成了。

下面展示了一些问题范例，可以用来制订宗旨：

如何向新客户介绍企业的目标呢？

你认为企业的目标是什么？

企业的主要作用是什么？

你如何描述企业的作用呢？

你如何阐述企业存在的最重要原因呢？

企业的主要着眼点是什么？

读者也许注意到了，其中一些问题意思似乎一样，只是表述不同。记住，表述措辞方面的技巧也可以运用到访谈提问中。向多人提出相同的问题会得

到不同的回答，因为每个人对问题的解读会有细微的区别。有时，对方只是久久地瞪着眼，似乎在说：“我连咖啡都没冲上呢。”先使用不同类型的措辞做实验，再决定哪种最合适。每个人组织和提出问题的方式可能和别人不同，但是这并无大碍，只要有适合自己的方法就行。

案例分析：为迈克自行车行制订宗旨

现在，需要为“迈克自行车行”制订宗旨了。在此之前，必须与车行业主开展访谈，收集他的业务信息。假定你有一位助理，名叫扎克瑞（Zachary）。他正在帮助你开展访谈。访谈将以如下的形式展开。

扎克瑞：“能告诉我，你为什么认为自己需要一个数据库吗？”

迈克：“我想我需要一个数据库记录我所有的存货。我也想记录所有的销售情况。”

扎克瑞：“我敢肯定数据库会解决所有这些问题。那么，你认为你的业务最重要的作用是什么呢？”

迈克：“为顾客提供各种自行车产品和相关服务。我们拥有许多大客户和老顾客。他们是我们最大的财富。”

（访谈将继续下去，直到扎克瑞问完所有问题清单上的问题。）

访谈之后，复查收集到的信息，明确宗旨。从与迈克的对话中可以得到一些答案，比如他需要跟踪产品、顾客和销售流程。但是，最有价值的地方是他对第二个问题的回答。你可以用其中的第一句作为基础，再兼顾访谈中确认的其他信息点，就可以将迈克的回答改写为宗旨，表述如下：

迈克自行车行数据库的目标是维护数据，支持零售业务和顾客服务运营。

当你确认宗旨已经初步成形，就和迈克一起讨论修改，确保他理解并认同数据库的目标。待双方都对此满意，就可以进入下一步，也就是明确任务目标。

明确任务目标

展开前面章节中的概述，任务目标就是指数据库中数据所支持的总任务。每个任务目标代表一个任务。这些任务目标提供的信息在整个数据库设计过程都会用到。例如，任务目标有助于明确表结构、字段说明、关系特征和视图。它们也有助于建立数据完整性和定义业务规则。最后，任务目标能引导开发方向，确保最终数据库结构符合宗旨。

优秀的任务目标

好的任务目标一般为一个陈述句，简单明了地定义总任务，不拖泥带水。多采用一般术语，语言简练、切中要点、准确明白。下列是一些目标任务的典型范例：

维护完整的患者地址信息。

记录顾客销售情况。

在任何给定时间内，确保一个客户代表负责的客户不多于20个。

记录汽车保养情况。

生成员工电话簿。

这些任务目标定义明确、简单易懂。每个任务目标表示一个总任务，表述清楚，简明扼要。例如，最后一个任务目标是生成员工电话簿，不过并未说明如何生成。不提及生成方式是因为这是应用开发过程的一部分。记住，任务的目标是明确数据库中的各种结构，引导数据库总体开发方向。

如果一个任务目标表示多个总任务，就应该予以分解。下列是一个不规范的任务目标示例：

我们需要记录所代理的艺人及其提供的服务类型，以及我们为艺人预定的活动。

以上目标任务存在两个问题：

1. **包含多个总任务。**显然，上述表达中有两个任务——记录艺人和记录预定安排。
2. **包含不必要细节。**不必提及艺人“提供的服务类型”。**服务类型**表示艺人的独特特征或者新任务。如果它是指艺人的独特特征，则应移除；否则，就应作为目标任务的基础。

改写这个目标任务需要删除冗余部分，分解为两个目标任务。（将冗余部分另外记录下来，留作后用。）下面是一种改法：

维护完整的艺人信息。

记录所有预定的活动安排。

现在，每个任务目标都只包括一个总任务，简单明确。设计数据库的时候，这样的任务目标易于使用。

制订任务目标

制订任务目标包括开展对用户与管理人员的访谈和基于访谈收集到的信息编写任务目标。

访谈的目的是确定数据所需支持的总任务类型。具体操作是向参与者提出开放式问题，并让他们尽量详细地回答。针对宗旨和任务目标的访谈是最简单的环节，因为通常每个人都会热心参与。（至少根据我的经验是这样。）让人们讨论他们的日常情况和对企业作用的看法是非常轻松的。这也是少数几次同时访谈用户和管理人员的机会之一；由于访谈的一般性质，两组人之间应有很多相同点。

记住，非常重要的一点是这里的访谈包括一般的讨论。讨论多是概念性的，而不是具体分析；这里的目的不是分析现有数据库或数据库应用，而是对数据库应支持的总任务做全面了解。记住，任务目标的一个目的是帮助引导数据库结构的开发。

开展访谈时，再次确认使用开放式问题提问。记住，开放式问题更易于让参与者做出更好的回答。提问应从参与者的日常工作、企业作用，以及他们认为数据库需要解决的问题这些方面入手。鼓励他们讨论工作和企业的各个方面。根据他们的回答，尽量把每一个回答记录成一个陈述句。这样，很容易就能将一句话转化为目标任务。访谈期间，可以提出下列问题：

你平常都干些什么？

你如何定义自己的工作描述？

你使用什么类型的数据呢？

你开出的报表是什么类型的呢？

你记录什么类型的事件呢？

你的机构提供什么类型的服务呢？

你如何描述自己的工作？

所有这些问题都可能换来参与者相当冗长的回答。这类问题的优点在于为后续问题提供了可能。例如，在提出上面最后一个问题之后，得到如下回答：

“首先，我尝试找出汽车的总体问题。接着，填写一个派工单并记录问题评估意见。最后，我将该汽车传递给下一环节的服务团队。”

你立刻就发现，这个回答有点冗长，但没关系。还应该意识到轻易就可以提出后续问题，比如：

“在你刚刚描述的环节中，是否涉及任何客户信息呢？”

即使回答是“没有”，参与者也仍然可以做出进一步的解释。这种后续问题也能刺激他的记忆，使他继续输出其他信息，这也许与初始回答的主题有关。

你可以从参与者的初始回答中得出以下一整套任务目标：

维护顾客汽车信息。

记录派工单。

维护服务团队信息。

维护修理工信息。

维护顾客信息。

其中三个目标是直接从回答中提取出来的。由于回答中明确表达了这几个目标的主题，所以很容易确定其内容。最后两个任务目标从基于回答所做的假设中衍生出来。这种技巧（可以当作“言外之意”）是经验丰富的数据库设计者们经常使用的。在制订任务目标时，你也应使用。这项技巧取决于识别的能力：回答中暗含的信息，以及明确传达的信息分别是什么。因此，保持精神集中，听出弦外之音。缺少好的假设，整体任务目标就显得不完整。

再来看看下列回答，判断其中是否存在隐含信息：

“我为客户提供娱乐业务，包括商业和非商业客户。非商业客户通常是个人或小团体，服务项目包括婚礼、生日、纪念日等。商业客户包括夜总会和企业等。夜总会提前六周预约服务；企业预约的服务包括公司聚会、产品发布以及各种宣传活动。”

除了其中传递的明确信息，至少还可以发掘两处隐含信息。第一处隐含信息涉及维护预定了参加活动的艺人信息。经纪人要清楚艺人姓名、电话号码、电子邮箱、档期安排，以及是否愿意去外地演出等。第二处隐含信息涉及维护演出活动本身信息。经纪人须了解与活动相关的所有细节，以确保演出顺利运作。

在制订任务目标时，一定要牢记隐含信息的重要性，设法把重要的隐含信息挖掘出来。

接下来是任务目标的“结语”：确保任务目标定义恰当明确，保证你和数据库使用对象都清楚其含义；从每位参与者的回答中找出所有隐含信息。

案例分析：为迈克自行车行制订任务目标

现在，是时候对迈克和他的员工进行访谈了。下列为迈克访谈的部分记录。同样，你的助手扎克瑞正在为你开展访谈。

扎克瑞：“能告诉我你想在数据库中记录什么吗？”

迈克：“嗯，可以。很简单，我想记录库存、顾客和销售。”

扎克瑞：“你认为还有什么和这些相关呢？”

迈克：“我想，如果我们要记录库存，就应该了解供应商。”

扎克瑞：“那每笔销售中的推销员呢？”

迈克：“啊，对。我们应该记录员工的信息。如果没别的什么要求，从人力资源的角度来处理这个问题应该不错。至少，我妻子是这么跟我说的！”

（访谈继续，直到扎克瑞问完所有问题。）

访谈完成之后，复查收集到的所有信息，制订合适的任务目标。制订任务目标时，要牢记“结语”。以下是迈克自行车行数据库的几个任务目标。

维护完整的库存信息。

维护完整的顾客信息。

记录所有顾客销售情况。

维护完整的供应商信息。

维护完整的员工信息。

初步完成一系列任务目标后，与迈克及其员工一起再进行复审。当他们表示理解其含义并感到满意时，任务目标也就完成了。将任务目标保存到文档中，留作后用。

小结

本章以讨论访谈过程开始，论述了访谈是数据库设计过程的一个重要部分以及开展优质访谈的重要性。读者从中了解到开放式问题和封闭式问题的区别，以及这两种问题适用之处。最后还介绍了一套访谈指南。读者从中应该领会到，使用这些方针有助于确保访谈富有成效并成功。

宗旨是接下来讨论的话题。在第4章“概念性概述”的基础上进行了延伸，着眼于宗旨如何阐述数据库的特定目标。这个过程包括开展访谈和了解企业，然后根据之前收集到的信息制订宗旨。文中还定义了优秀的宗旨所具备的特征。优秀的宗旨为设计指明了焦点。

随后，我们讨论了任务目标，也是在第4章基础上做了拓展。任务目标表示在数据库中的数据上可执行的总任务。在明确宗旨之后，就要制订任务目标。我们接着探索了如何制订任务目标。开展与用户和管理人员的访谈，将从中收集到的信息作为每个任务目标的基础。还讨论了优秀任务目标的特征。定义简单明白的任务目标有助于明确数据库中各种结构。

思考题

1. 访谈为什么重要？
2. 对许多人访谈时，会遇到什么问题？
3. 对用户和管理人员分别访谈的主要原因是什么？
4. 判断题：访谈中常用到封闭式问题。
5. 你应该设法让参与者做出什么样的回答？
6. 访谈最重要的指南是什么？
7. 宗旨是什么？

8. 指出优秀宗旨的两个特征。

9. 判断题：制订宗旨时无须了解企业。

10. 怎样才算完成宗旨？

11. 怎样才算是一项任务目标？

12. 说出优秀任务目标的两个特征。

13. 判断题：为了制订任务目标，应该采访用户和管理人员。

14. 员工日常工作和任务目标有何关系？

15. 判断题：一个任务目标能描述多个任务。

16. 指出从回答中得出任务目标的两种方式。

17. 怎样才算任务目标完成？

维护完整的所有信息。

维护完整的顾客信息。

记录所有顾客销售情况。

维护完整的。

维护完整的员工信息。

初步完成一系列任务目标。

公司量温案。

第 6 章

分析现有数据库

要看清眼前的事物，需要进行一番持续的争斗。

——乔治·奥威尔

In Front of Your Nose

本章内容

了解现有数据库

开展分析

了解如何收集数据

了解如何呈现信息

开展访谈

用户访谈

管理人员访谈

编辑完整字段列表

案例分析

小结

思考题

了解现有数据库

欲知路向何方，必先知身在何处。

这句格言诠释了本阶段暗含的全部哲学。必须投入一定时间将数据库了解清楚,才能:

- 判断数据库是否满足当前机构信息要求。
- 发现现有结构缺陷。
- 决定数据库该如何进化,从而满足将来机构信息要求。

可以将现有数据库当作开发新的数据库的资源。不过,必须谨慎判断现有数据库的哪些方面仍然有用,哪些方面应该舍弃。通过回答下列问题,就可以做出判断:

机构使用的是哪些类型的数据?

机构如何使用这些数据?

机构如何管理和维护这些数据?

这些问题的答案能提供重要信息。借助它们,就能设计最适合所在机构的数据库。

通过分析机构现有数据库,就能得到这些问题的绝佳答案。机构很有可能正在使用的数据库与以下几类有关。

- **纸质数据库 (paper-based database)**, 也称为档案系统 (file system) 通常包含各种手写或打印的文档。这些文档存放在档案夹中或合订成册。档案夹或册子则借助某种编码机制 (例如, 独有的编号或彩色标签) 标识并存入档案柜。根据数据库的规模不同, 这些档案柜也可能设定有编号。
- **遗留数据库 (legacy database)** 是指存在和使用数年以上的数据库。它包含存在于主机、网络服务器以及个人电脑中的各种数据结构和用户界面。这些结构和界面的性能、功能和效能都与开发者的技能和知识、应用开发工具以及所使用的数据库管理软件密切相关。
- **人类知识库 (human-knowledge base)** (广泛而言) 是基于机构内部一个或多个员工的记忆建立起来的。这些个人拥有相应机构某些方面的知识 (比如, 顾客信息或产品细节), 对机构开展业务起到关键作用。

分析的目的是判断机构所使用数据的类型、机构管理和维护数据的方式，以及机构查看和使用数据的方式。如果这一调查处理得当，就可以为定义初始字段和表结构节省时间。

在分析时，要评审机构收集和呈现数据的各种方式，并对用户和管理人员开展一系列访谈。然后，使用收集到的信息定义初始字段列表，以及确定应该包括在初始数据库结构中的表。如果分析揭示当前数据库设计不当，可以采取预防措施，确保在新数据库中不再犯相同错误。无论当前数据库具有怎样的缺陷，都仍有助于判断新数据库中应该包含的一些字段和表。

分析当前数据库时，你首先应谨记以下规则：

切勿将当前数据库结构用作新数据库结构。

遵循这一规则将有助于避免不必要的错误，让设计收到最大的成效。

在分析的时候，数据库开发的新手常常停下来想（经验丰富的老手有时也会这样），“这个数据库似乎不赖。不如就此停止，直接使用这个数据库建立新数据库。”这个想法尤其坏事，因为当前数据库结构中隐含的每一个问题都会复制到新数据库中。这些问题包括设计不当的表结构、定义不当的关系以及字段规范不一致；它们迟早会暴露出来。因此，应该遵循前面所述规则，避免这种风险。一定要记住，明确定义新数据库结构始终要比复制现有结构好。毕竟，如果原有数据库无懈可击，就不会建立新的数据库了。

在这一阶段，通常是分析纸质数据库和遗留数据库。许多机构在一定程度上同时使用了这两种数据库，人们使用相同的分析过程对其进行处理。诚然，纸质数据库和遗留数据库的分析方式有些许差异，但是这些差异更多源于数据库本身，与总体分析过程关联不大。不过，无须为此担心，因为笔者已经将它们完美地融入了本书所展示的分析过程中。

纸质数据库

纸质数据库包含使用纸笔记录、存储和维护的数据，它们形式多样，大小不一，构造也不尽相同。一些较为常见的格式包括索引卡、手写或打印报

表,以及各种预印表格。在办公室工作的人们都对这种数据库非常熟悉。

读者会发现,分析纸质数据库是一项非常艰巨的工作。首要任务就是找到完全理解数据库运作方式的人,从他那里了解其用途和目的。数据库本身就存在多个问题,特别是在数据收集和管理方式方面。这种数据库通常包含不一致的数据、错误数据、重复数据、冗余数据、不完整的条目,以及早就应该清除的老数据。显然,分析这种数据库的唯一目的是找出可以融入新数据库的项。例如,可以从原数据库中一个表格的各个部分抽取单个数据片段,将它们转化为新数据库的字段。

遗留数据库

遗留数据库是指存在和使用至少达五年的数据库。主机数据库和较旧的个人电脑上建立的数据库通常被归为这一类。此类数据库名称中使用“遗留”一词有多个原因。第一,它表明数据库已存在很长时间,可能已经超出人们清晰记忆的范围。第二,遗留一词意指数据库原始创建者已经调换了职位或者公司,因此数据库成了创建者遗留下来的东西。第三,它暗示可能无人完全了解数据库结构或者它在 RDBMS 应用程序中的实现方式。

在分析过程中,主机遗留数据库展现出一些特殊的问题。其中一个问题就源于多个陈旧主机数据库基于层次或网状数据库模型建立。如果机构内部无人真正了解这些模型,就需要花费一定时间破译数据库的结构。在这种情况下,将每个数据库结构中的数据都打印出来会非常有用。

即使遗留数据库根据关系模型建立,也不能完全保证结构完善。不幸的是,许多人未能完全理解关系数据库的概念,却创建这样的数据库。(阅读完本书,你就不会犯这种错误。)结果,许多旧数据库或结构不当,或效率低下。

大量个人电脑上运行的遗留数据库设计不当或低效。其中许多都是在非关系数据库管理系统中开发和实现的,这意味着它们不具备关系模型的优势。这种数据库通常都有重复字段和冗余数据。在下文中,你将认识到这些问题会严重危害到数据完整性。

分析遗留数据库比分析纸质数据库要简单一些，因为遗留数据库通常更为组织化、结构化，数据库内部结构定义也较为明确。另外，人们往往使用应用程序与遗留数据库中的数据交互。（应用程序具有一定价值，因为它能揭示大量信息，与数据结构和使用遗留数据库中数据所能执行的任务有关。）执行分析所花费的时间一定程度上取决于平台（主机或者个人电脑）、实现该遗留数据库所用 RDBMS，以及应用程序。

关键要记住无论是分析纸质数据库还是遗留数据库，都应该有耐心，有条不紊地完成整个过程，确保分析准确、全面。

开展分析

分析过程分为三个步骤：审核数据收集方式，复查信息呈现方式，以及开展用户和管理人员访谈。

进行前两个步骤时，需要与公司内部各个人群进行对话。确保对话只涉及正在进行的评审。之后，还有机会向他们询问更深入的问题。记住，这些评审工作是后续访谈的重要准备工作。这些评审工作有助于你确定在访谈中需要问到的问题。

了解如何收集数据

分析过程的第一步是审核数据收集方式，包括从索引卡和手写或打印报表，到预印表格和数据输入界面（例如数据库程序或网页浏览器所使用的数据）的所有数据。

首先审核所有纸质文档。找出机构使用哪些类型的文档记录数据，每种文档各收集一个样本。将这些样本整理并保存到一个文件中，留待后用。例如，一些机构使用索引卡收集供应商的数据。查找所有索引卡，直到发现条目最为完整的索引卡。之后，复制该样本，保存在相应文件夹中。对每种文档都执行这一过程。图 6.1 展示了机构如何使用纸质文档收集数据的两个示例。

Al Office Supplies	
Suite 133	
7739 Alpine Way SE	
Seattle, WA 98115	
Susan Black	519-5883
Email	susanb@A10S.com

Employee Fact Sheet			
Name: George Chavez		Date Hired: June 30, 2009	
Address: 7527 Taxco Drive		City: Seattle	State: WA Zip: 98115
Phone: 553-0399	Date of Birth: 09/22/85		SSN: 987-65-0049
Education:			
Name of Academic Organization		Location	Year Graduated
University of Texas at El Paso		El Paso, TX	2007

图 6.1 收集数据所用纸质文档示例

接着，评审机构收集数据使用的所有计算机程序。此处的目标是收集一组截屏样本，代表机构通过这些程序使用数据的方式。提醒一句：各人对收集和管理数据都有自己的心得，知道如何巧妙地使用常见的程序，比如文字处理软件和电子数据表格。确保与机构内部熟悉计算机使用的人员交谈，搞明白机构使用哪种程序管理数据。

在评审每个程序时，找出最能代表程序收集数据的界面。所找出的界面应如图 6.2 所示。

第一个界面通常出现在数据库程序中，第二个界面则出现在电子数据表格程序中。一旦找到合适的样本，用截屏软件截图并粘贴到文字处理程序的文档中，并标注源程序的名称和创建该截屏的日期，然后打印该文档。接着，继续审核程序并酌情重复上述过程。然后对每个程序都重复这一完整过程。待所有合适的截屏副本打印完后，将它们收集到一起，保存到一个文件夹，以待后用。

Mike's Bike Shop

File Edit View Insert Format Records Tools Window Help

Customer Information

Name (F/L): John Holmes

Address: 725 Hunter Place

City: El Paso

State: TX Zip: 79915

Status: Active

Phone: 778-9715

Email: johnh@am.com

< Back Next > Edit Save

Mike's Bike Shop - Product Information

File Edit View Insert Format Tools Data Window Help

	A	B	C	D	E
1	Product ID	Product Description	Category	SRP	Qty On Hand
2	9001	Shur-Lok U-Lock	Accessories	75.00	
3	9002	SpeedRite Cyclecomputer		65.00	20
4	9003	SteelHead Microshell Helmet	Accessories	36.00	33
5	9004	SureStop 133-MB Brakes	Components	23.50	16
6	9005	Diablo ATM Mountain Bike	Bikes	1,200.00	
7	9006	UltraVision Helmet Mount Mirrors		7.45	10

图 6.2 典型数据库界面和典型电子数据表格界面

接下来, 检查该机构通过互联网收集数据所用的网页。需要找出的页面与数据库应用程序中的数据输入表单十分相似。图 6.3 所示为这种网页的一个示例。

Reply Form - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address: http://www.mikesbikes.com/reply.htm

Reply Form

If you'd like to obtain a brochure of our services, please fill out the form below and press the "Submit" button when you're finished.

First Name: Earl Address: 7402 Kingman Drive

Last Name: Morgan City: Seattle

Email Address: earlm@datatexcg.com State: WA Zip Code: 99125

Comments:

Please send me a brochure at your earliest convenience.

Submit Cancel

图 6.3 典型网页式数据输入界面示例

这里可以遵循与应用程序所使用的一样的检查过程。把给定网页的截屏粘贴到文字处理文档中,标注程序名称和界面抓取日期,打印出来。继续审核网页,酌情重复这一过程。待所有合适截屏都打印完后,即将它们收集到一起,存放到一个文件夹中,以待后用。

确保为包含分析时收集到的样本的文件夹做好标记。在后续设计过程中使用这些材料时,组织材料所花费的少量时间将收获大回报。

了解如何呈现信息

分析的第二步是审核机构将数据作为信息呈现的各种方式。此过程将审核多个项目,比如手写文档、计算机打印件、界面演示,以及网页。

下列为最常见的三种呈现方法。

1. **报表 (report):** 报表是以受众接受为目的,整理和呈现数据的所有文档(手写文档、打印文档,以及计算机生成文档等)。尽管使用文字处理器、电子数据表格,或者其他软件程序是生成报表的标准方法,但是你还是会发现一些手写的报表。
2. **幻灯片 (slide show, 又名界面演示 [screen presentation]):** 这种幻灯片由一系列讨论各种主题的界面组成,它们以一种组织化的方式排列。幻灯片通常借助程序创建,比如 Microsoft PowerPoint 或 Corel Presentations,在计算机上执行。另外,还可以使用透镜式投影仪把一系列塑料片投放在屏幕上。(本文假设的情况是计算机幻灯片。)
3. **网页 (web page):** 许多机构通过网页在网站上存储了大量有效信息。网页的使用和报表非常相似。的确,它不过是另外一种报表。

首先,识别和评估机构的数据库生成的每份报表,不论是手写报表还是应用程序生成的报表。收集报表的样本,并如前面步骤中一样,将之保存在文件中。总的说来,在这个步骤中执行这个任务要更为容易,因为机构内部人员通常比较熟悉自己使用的报表。报表的副本一早就已具备,如有必要,

大多数报表都可以重印。图 6.4 所示为一份手写报表和一份文字处理程序生成的报表。

接下来，审核使用或包含数据库中数据的幻灯片。不必评审所有幻灯片，但的确需要审核对数据库中数据有直接影响的内容。例如，某新产品未从数据库中提取任何数据，则无须对该幻灯片进行评审。另一方面，销售统计的幻灯片包含数据库数据，则需要评审。

一旦确定需要评审的幻灯片，即对每个幻灯片仔细评审，并为使用或包含数据库数据的幻灯片截屏。复制截屏并保存到文字处理文档中，打印该文档，再将文档存放在文件夹中，留待后用。（文件夹应标注幻灯片名称和截屏日期，以备不时之需。）每个幻灯片均须遵循这一过程。避免不小心将多个幻灯片混在一起，因为这一错误必定会导致混乱和更为严重的后果。

Employee Phone List as of 05/16/11				
John Alcot	554-3002			
Regina Allen	752-5593			
George Chavez	623-3292			
Ryan Boyd	554-2991			

Current Product Inventory				
Product ID	Product Description	Category	SRP	Quantity
9001	Shur-Lok U-Lock	Accessories	75.00	
9002	SpeedRite Cyclecomputer		65.00	20
9003	SteelHead Microshell Helmet	Accessories	36.00	33
9004	SureStop 133-MB Brakes	Components	23.50	16

图 6.4 手写报表和计算机生成的报表

图 6.5 所示为需要评审的幻灯片示例。

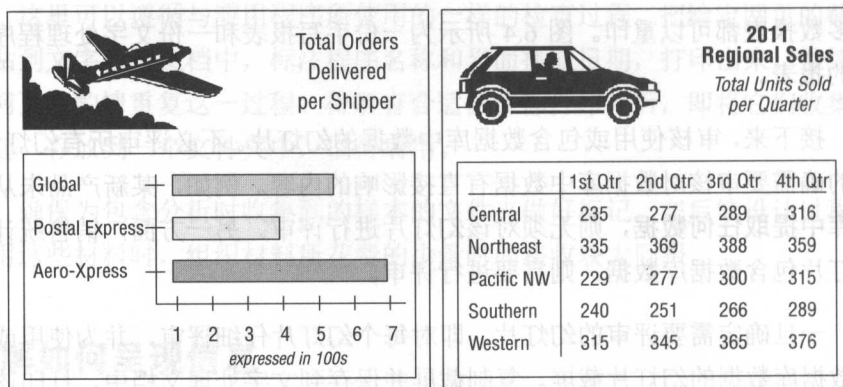


图 6.5 幻灯片示例

在一定情况下, 评审幻灯片很困难, 决定是否将某一幻灯片选作样本也需视情况而定。因此, 与最熟悉幻灯片的人士密切合作, 确保样本中包含所有合适的幻灯片。

最后, 评审直接从数据库中提取信息的网页。其操作过程与评审幻灯片一样。如前所述, 你需要评审与数据库中数据直接相关的网页。例如, 可以排除介绍机构历史的网页, 而对于展示区域员工信息的网页, 就需要加以评审。

识别完需要评审的网页后, 为每个网页截屏。将截屏复制到文字处理文档中, 打印该文档, 然后保存到文件夹中, 留待后用。(在文档中, 每个截屏下都需标注 URL 地址和当前日期, 以备不时之需。)

图 6.6 所示为需要评审网页的示例。

与创建和开发机构网站的人员合作。他们能指出具体需要评审的网页, 为你节省时间。



图 6.6 图中网页呈现了数据库中信息

开展访谈

至此，读者已经对机构如何收集和呈现数据有了大致的了解。接下来应该对用户和管理人员开展访谈，决定机构如何使用数据。在分析阶段，访谈有用的原因如下：

- 为之前评审阶段收集的样本提供细节信息。在前面阶段中，与用户和管理人员的讨论仅仅是为了确认（从一般的角度）机构如何收集和呈现数据。然而在这一阶段，读者将针对所收集的样本提出细节问题。这确保你能弄清楚特定样本的方方面面。
- 提供关于机构使用数据方式的信息。这些访谈将让你了解用户平常如何使用机构的数据，以及管理人员如何使用基于数据的信息，管理机构事务。
- 有助于定义初始字段和表结构。这一阶段用户和管理人员做出的回答将帮助确定初始字段和表结构。
- 有助于定义未来信息要求。与用户和管理人员讨论机构未来发展很可能揭示出数据库必须支持的新型信息要求。

毋庸置疑，访谈对数据库最终结构和数据库设计圆满成功有着重要影响。唯有全面彻底的访谈才能确保数据库满足机构信息要求。

基本访谈技巧

为了使访谈成功，首先必须学习一些基本访谈技巧。本文为此提供了一套基本技巧，对于数据库设计过程中的每次访谈都可以使用。这些技巧相对简单易学、使用方便，它们能让你获得任务所要求的信息。

刚开始学习时，必须严格按照要求进行实践。等到访谈经验丰富之后，就可以灵活运用。开展访谈是一门技巧性工作。和其他许多类似的事情一样，付出一定的耐心和努力之后，就能达到一定的水平。

问题的重要性

学习如何提问是一项有价值的技巧。如果想要设计数据库取得成功，就必须掌握它。正是使用这项技巧，你才能了解你的（或客户的）业务如何运作，并且收集到开发数据库结构所需的信息。此外，如你所料，这也是整个设计过程中开展访谈都需要的一项技巧。这看似空口说白话，但是绝非夸大其词。

访谈过程

在访谈中，同时使用开放式和封闭式问题。随着访谈的深入交替使用这两种问题。开放式问题本质上更为普遍，让你能够聚焦具体主题。反之，封闭式问题更为具体，让你能够聚焦某一主题的特定细节。例如，采用开放式问题开始，为讨论确立若干一般主题，然后选取其中一个，询问与之相关的具体（封闭式）问题。你可以向一位参与者提出如下开放式问题：

“你能说说自己日常工作是干什么吗？”

大多数参与者会使用三个或更多的句子来回答这类问题。如果某参与者能向你提供详细具体的回答，就再好不过了。因为这种回答更易于处理。为说明这一点，假设该参与者以下列方式回答：

“作为一名客户代表，我负责 10 位客户。我的客户都是通过预约来到展厅，了解我们当季提供的产品。我的部分工作是为他们解答产品方面的问题，根据受欢迎的选项进行推荐。一旦他们决定好将购买的产品，我就为该客户写下销售订单。然后，我将销售订单交给我的助理，由他迅速完成订单并发送给相应客户。”

上述回答非常精彩。这位参与者不仅回答了问题，而且为你提供了提出后续问题的机会。他的回答也暗示了随后将在访谈中讨论的多个主题。

❖注意：简练的回答，比如“我负责填写顾客销售订单”，传达的信息很少，所以必须设法让参与者畅所欲言，从而详细地了解这个过程。回答过于简短通常意味着参与者处于紧张拘束的状态。在这种情况下，你可以暂时聊聊一些不相关的话题或者让其选择更为熟悉或轻松的话题作为开场，这样他就会放松。

确定主题

提出开放式问题，确定回答中暗含的主题。可以通过从回答的句子中寻找名词确定主题。主题往往由名词表示，意指一个人、一个地方、一个物体或者一个事件（在给定时间点发生的事件）。不过，一些名词表示的是人、地方、物体或事件的特征；但你不必为此担心。因此，确保自己只关注专指人、地方、物体或事件的名词。（注意，每个名词只标记一次。）如果确定了某个名词，即用双下画线标记，这样就能确保记录下了所有需要讨论的主题。示例如下：

“作为一名客户代表，我负责 10 位客户。我的客户都是通过预约来到展厅，了解我们当季提供的产品。我的部分工作是为他们解答产品方面的问题，根据受欢迎的选项进行推荐。一旦他们决定好将购买的产品，我就为该客户写下销售订单。然后，我将销售订单交给我的助理，由他迅速完成订单并发送给相应客户。”

确定所有回答中的名词之后，将它们列在纸上；这就成为了你的主题列表（list of subjects）。随着设计过程的推进，你将添加更多的主题。编辑这

个列表要仔细、有条不紊，因为随着访谈的深入，你将用它来指导进一步的讨论，帮助你在后续过程中定义表。

下列是从前面回答中得出的主题：

客户代表 工作

预约 产品

助理 销售订单

客户 当季

项目 展厅

接下来，你可以将这个列表当作下一步提问的基础。

❖注意：下文将这整个过程称为确定主题技巧（Subject-Identification Technique）。

理解这些名词在回答中的含义，就可以验证它们是否真正的主题。例如，“客户代表”取自第一句话中，你可以根据其含义假设主题表示对象（人、地方，或物体）。“预约”取自第二句话，你可以根据其含义假设主题表示一个事件（某个时间点上发生的事情）。

确定特征

确定回答中暗含的主题后，选择其中一个主题并提出与之相关的后续问题。使用这一连串的问题，尽量获取该主题相关的更多细节。确保后续问题更加具体。后续问题的性质取决于参与者的回答。例如，基于文中的示例，你可以继续询问销售订单的具体问题，或者开始新一轮与客户相关的问题。假设，为了更多地了解销售订单，提出如下问题：

“我们不妨讨论一下销售订单吧。为客户完成销售订单需要干些什么？”

注意，这个问题一开始就点明了聚焦的主题。选定讨论主题后，就应该

使用这种技巧来进行引导。还要注意，这个问题是开放式的，鼓励参与者提供与主题相关的细节，并且可以为参与者接下来的回答确立焦点。

现在，假设参与者做出如下回答：

“嗯，我首先输入所有客户信息，比如客户姓名、地址、电话号码，以及电子邮箱。然后再输入客户想要采购的项目。待所有的项目输入完后，计算总和。这样就完成了。哦，我还忘了说：如果客户有传真和配送地址，就也要输入进去。”

使用确定主题技巧分析这一回答，确定回答中是否暗含那些主题。然后将新的主题添加到主题列表中。记住，列表只收录表示人、地方、物体或事件的名词。

确定新主题之后，寻找与讨论中的主题相关的细节。目标是尽量获得该主题的更多详情。现在要搜寻表示该主题特征的名词，这些名词描述的是该主题的某些方面。这些名词轻易就能发现，因为它们通常为不加修饰的名词（“电话号码”、“地址”）。相反，确定主题的名词通常带有从属关系（“客户电话号码”、“公司地址”）。

尽量描述更多主题的特征。使用单下画线标出表示主题特征的名词，示例如下：

“嗯，我首先输入所有客户信息，比如客户姓名、地址、电话号码，以及电子邮箱。然后再输入客户想要采购的项目。待所有的项目输入完后，计算总和。这样就完成了。哦，我还忘了说：如果客户有传真和配送地址，就也要输入进去。”

在回答中找出合适的名词后，将这些名词记录在纸上；这就是特征列表（list of characteristics）。在后续过程将在列表中添加更多的特征。在确定数据库字段时将用到这一列表。使用另外的纸记录特征列表。切勿将主题列表和特征列表混在一起！（在第7章“建立表结构”中要为数据库定义表，届时就会明白为什么需要放在不同的列表中。）

下列为从之前回答中得出的特征：

地址	姓名	总和
电子邮箱	电话号码	
传真	配送地址	

上面的列表组成了讨论中的主题的特征列表。这些特征最终会变成数据库中的字段。

❖ 注意：下文将这整个过程称为确定特征技巧 (Characteristic-Identification Technique)。

根据这些名词在回答中的含义，验证它们是否真正的特征。例如，“姓名”出自第一句话中，你可以根据其含义，假设它描述的是主题“客户”的某个方面。“配送地址”出自最后一句话，你可以根据其含义，假设它也表示主题“客户”的某个方面。

讨论完特定主题后，继续讨论主题列表上的下一个主题，并开展相同模式的提问。首先以开放式问题开始，确定回答中暗含的主题，随着讨论的推进提出更具体的问题，以及尽量找出更多的主题特征。有条不紊地继续这个过程，直到讨论完列表中所有的主题。

应该将确定主题技巧和确定特征技巧尽可能学透，因为在与用户和管理人员访谈以及为初始数据库结构确定字段和表期间都会用到这些技巧。注意，如果操作熟练，则可不使用单下画线和双下画线。最终，等到经验足够丰富，就可以在头脑中灵活运用这些技巧。

开始访谈之前

在对用户和管理人员的访谈中，可以运用上一节中学到的技巧。两组访谈的唯一区别在于主题和问题内容。

访谈过程包括两组讨论：一组是用户访谈，另一组是管理人员访谈。首先开展用户访谈，因为他们工作在“第一线”。他们最清楚机构日常运作方面的细节。而且，从用户处收集到的信息有助于理解管理人员的回答。

用户访谈

访谈的第一部分是开展用户访谈。访谈主要围绕四个问题展开：

1. 用户当前使用的数据类型。
2. 用户当前如何使用数据。
3. 前面两个分析环节收集到的样本。
4. 用户日常工作中要求的信息类型。

因为这些问题都是以数据和信息为中心，所以必须确保自己理解并且始终牢记数据和信息之间的区别。第3章“术语”中已经提到，数据是存储在数据库中的值，而信息则是以使用或查看为目的经处理后的数据。牢记这些定义能够确保合理看待每个问题以及访谈各个环节都取得成功。

评审数据类型和用途

如果在访谈之先，对问题做好了充足的准备，一般就能同时讨论前两个问题。这部分访谈的目标是确定用户当前使用的数据类型以及如何使用这些数据支持工作。在后续的设计过程中，将使用这些信息，帮助定义字段和表结构。使用数据收集和数据表述样本有助于制定用户数据方面的问题。（不过，应该分别处理这两方面的样本。）讨论期间，首先提出开放式问题，确定回答中的主题，然后提出更为具体的后续问题为每个主题确定特征。

访谈开始时，向每个参与者询问其日常工作内容。等到某位参与者介绍了整体工作内容后，再要求他描述更多工作的细节。也许，他能带你领略一番他的工作流程。

下列为访谈中一段常见的对话。

采访者：“你日常都干什么？”

参与者：“各行各业的人们向我提交土地使用申请，我再将这些申请进

行注册，并和听证审查官一起商定听证日期。如果申请人对申请有任何疑问，我也会为他们解答。”

采访者：“我们来聊聊申请吧。申请涉及哪些方面呢？”

参与者：“实际上，有多个方面。包括申请的名字和类型、名称和地址，以及位置。”

采访者：“说说与申请类型和名称相关的方面吧。”

参与者：“我们记录四个方面：申请类型、区域名称、项目目的以及项目描述。”

注意，采访者首先使用开放式问题。参与者做出回答之后，采访者使用确定主题技巧从回答中找出主题。然后选定主题，使用另一开放式问题将参与者注意力集中于该主题。因为参与者接下来的回答本质上是笼统的，采访者聚焦于该主题的特定方面，并继续提出更为具体的问题引导参与者给出详细的回答。

随着讨论的深入，采访者可以继续缩小问题的范围。参与者回答问题的同时，采访者也可以继续使用确定特征技巧找出回答中出现的主题特征。确定所有主题特征之后，采访者就可以顺势提出下一主题，重复相同的过程，直到完成所有列表中的主题。读者作为采访者时，也需要按照上述过程进行操作。

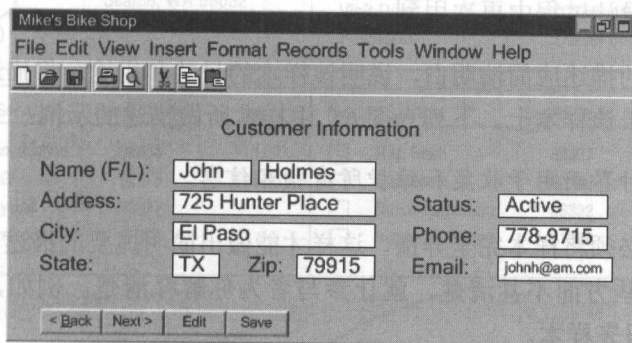
❖注意：本书中的对话示例都经过设计，较为简单；访谈过程中讨论的所有问题也是如此。它们只是展示特定技巧或技术所借用的工具。因此，不必在对话或问题上过多纠缠；而应该注重讨论是如何例证相应技巧或技术的。从这一角度来看，示例将更为有效。

评审样本

接下来一轮讨论的重点是从前面分析过程中收集到的所有样本。其目标是确定如何使用样本所表示的对象，弄清不清楚的方面，以及为每个样本提

供描述。

既然已经对参与者日常使用的数据有所了解，那么与参与者谈论样本就相对简单了。首先就特定样本提出问题。图 6.7 所示为数据收集样本的一个实例，你可以把它作为开端。



The screenshot shows a software window titled "Mike's Bike Shop" with a menu bar (File, Edit, View, Insert, Format, Records, Tools, Window, Help) and a toolbar. The main area is titled "Customer Information" and contains a form with the following fields and values:

Name (F/L):	
John	Holmes
Address: 725 Hunter Place	
City: El Paso	Status: Active
State: TX	Phone: 778-9715
Zip: 79915	Email: johnh@am.com

At the bottom of the form are four buttons: "< Back", "Next >", "Edit", and "Save".

图 6.7 数据收集样本

❖注意：上文对话示例所做描述也适用于此图以及下文中其他图示。这些图都只是展示特定技巧或技术的工具。使用相同的方法对读者更为有利。

先回顾访谈之前的讨论所记下的笔记，再提出问题。读者需要确定已经讨论的话题与将要讨论的样本相关联。例如，在之前的一次讨论中，一位参与者提到他的部分工作是记录所有顾客。将这句话作为开端，就可以问他如何使用这种特定的数据收集样本执行该任务。

“在之前的一次讨论中曾提到你负责记录所有顾客。那么，这种界面如何帮助你完成任务呢？”

这个问题安排巧妙。它一开始即聚焦于特定的主题，然后将参与者的注意力集中转移到样本上来。问题是开放式的，参与者会给出清楚完整的回答。

现在，假设参与者回答如下：

“使用这种界面可以输入新顾客，以及修改和维护现有顾客的

所有信息。”

如果你满意这一回答，那就把它作为描述该样本的基础。另一方面，如果这一回答并未使问题得到全面解答，则继续提出一系列合适的问题，直到参与者将样本的目的和用途说明清楚。必须为所有样本提供描述，因为它们将在后续的设计过程中再次用到。

对样本的描述应简洁明白，说明该样本的目的和用途。将描述用纸条记录，并附注于该样本上。下列为图 6.7 中样本所做描述的示例：

这种界面用于收集和维护所有顾客信息。

因此，必须将样本完全弄懂，这样才能做出简明扼要的描述。如果对于某样本的一些方面不甚清楚，就让参与者为你解释清楚。例如，假设使用图 6.8 所示报表样本。


Current Product Inventory				
Product ID	Product Description	Category	SRP	Quantity
9001	Shur-Lok U-Lock	Accessories	75.00	
9002	SpeedRite Cyclecomputer		65.00	20
9003	SteelHead Microshell Helmet	Accessories	36.00	33
9004	SureStop 133-MB Brakes	Components	23.50	16

图 6.8 报表样本

如果你不清楚缩写“SRP”的含义，一定要让参与者予以说明——切勿随意猜测。如果猜测被证明不实，就会浪费更多宝贵的时间和精力。

对每个样本做出描述时，也许会发现难以为复杂的样本给出描述。如果一个样本表示多个主题，则该样本为复杂样本。例如，图 6.8 所示样本只包含一个主题：产品。然而图 6.9 所示样本包含至少 3 个主题：医疗服务、护理服务，以及患者。对于这种情况，往往需要付出更多的努力来确定复杂样

本的目的和用途。在某些情况下，你必须使用确定主题技巧识别所表示的主题。

 Eastside Medical Clinic 7743 Kingman Dr. Seattle, WA 98032 (206) 555-9982			Patient Name: George Edelman Patient ID: 10884 Visit Date: 02/16/12 Physician: Daniel Chavez		
---	--	--	---	--	--

Doctors Services	Service Code	Fee	Nursing Services	Service Code	Fee
<input checked="" type="checkbox"/> Consultation	92883	119.00	<input type="checkbox"/> R.N. Exam	89327	
<input checked="" type="checkbox"/> EKG	92773	95.00	<input type="checkbox"/> Supplies	82372	
<input type="checkbox"/> Physical	98377		<input type="checkbox"/> Nurse Instruction	88332	
<input type="checkbox"/> Ultrasound	97399		<input type="checkbox"/> Insurance Report	81368	

图 6.9 复杂报表样本示例

不妨假设，读者使用的是图 6.9 所示报表样本，并且想知道护理服务方面的问题。比如，该机构是否使用这份报表间接维护当前护理服务列表。参与者仅仅回答是或否对你帮助并不大，所以你需要使用开放式问题获取更多信息。可以先提出下列问题：

“除这个样本上所列的之外，你们还提供什么护理服务呢？”

这种问题让参与者可以提供更为详细的回答；此外，你还可以根据其回答提出后续问题。继续这一示例，比如你得到如下回答：

“我们为特殊患者提供各种专门服务。你看，这份报表上只有一般的服务。不过，我可以让你看看完整的服务列表，它由凯瑟琳保存在她的电脑里。”

如果这一回答提供了清楚的解答，你从中明白了这个报表样本的目的，就可以进入下一环节；否则，继续后续问题，直到你满意为止。

评审信息要求

与用户讨论的最后一个问题是他们的信息要求。这一讨论的目标是了解个人用户是否会收到不直接受其控制或维护的数据信息、他们需要的附加信息类型，以及他们预测到未来需要的信息类型。在之后的设计过程中，将使用这次讨论中收集到的信息，帮助定义及验证字段和表结构。也可以用这些信息确认自己在前面的讨论中是否忽略了某些问题。

当前信息要求

用户一般通过各种报表接收需要使用的信息。因此，开启讨论最好的方式是评审报表样本。不过，这一次你更应该关注报表所依据的数据，而不是如何使用报表。通常而言，在用户接收的所有报表中，有一部分报表的信息不是基于他亲自创建和维护的数据。在这种情况下，必须判断数据源，从而确定该用户使用的所有数据，包括直接使用和间接使用的数据。

从报表样本中选取一份报表，和一位参与者共同决定产生这份报表所使用的数据。询问他是否参与创建和维护该报表所依据的数据。如果他给出肯定回答，则继续下一个样本；如果他回答没有，则需要确定数据来源。下面将展示这一过程。

假设，你有一位助手，名叫基拉。她与一位名叫琼的参与者就图 6.10 所示的报表样本展开讨论。

Customer Phone List			
Customer Name	CustomerType	Last Purchase	Phone Number
Alastair Black	Preferred	11/21/11	551-0993
Dave Cunningham	Silver	12/19/11	533-9182
Zachary Ehrlich	Preferred	11/16/11	515-3921
Bill Champlin	Gold	01/22/12	552-3884

图 6.10 样本报表

基拉先引出对话，琼随后提到她在电话营销部门工作。当基拉问及样本报表时，琼表示自己每周日早上都会收到。于是，基拉提出如下问题：

“是否由你提供生成这份报表所使用的数据呢？”

她接下来的行动取决于琼的回答。如果琼回答“是”，基拉可以继续讨论下一个样本；不过，基拉最好再提出一个问题确定琼回答的准确性。

“你每天亲自输入和维护这些数据吗？”

如果琼仍然回答“是”，基拉就可以放心地继续下一个样本了。

另一方面，如果琼开始给出否定回答，基拉就需要再问几个问题。首先，她要问琼是否为该报表贡献过任何数据。如果她给出肯定回答，基拉就要确认琼提交了哪些数据。然后，基拉询问琼是否知道其他数据的来源。

继续这个示例，假设琼开始给出否定回答，并且之后的对话如下：

基拉：“那么，你能跟我说说你为这份报表贡献数据了吗？”

琼：“我确实提供了客户姓名和电话号码。”

基拉：“顾客类型和最后购买日期不是你提供的，是吗？”

琼：“对。”

基拉：“你能说说是谁提供的吗？”

琼：“我不太确定，不过……”

基拉：“你清楚这些项目的来源吗？”

琼：“事实上，我清楚。它们来自销售部门。”

基拉：“谢谢，我要先将这一信息记在这个样本上，然后我们就可以讨论下一个样本了。”

注意，对话开始时，基拉首先确定琼是否为报表提交了数据。当琼说明自己为报表贡献了两个项目时，基拉接着就提出了一个问题验证琼没有提交

其他数据。最后，基拉为了进一步确认，又询问琼是否知道数据的来源。在这种情况下，只用了两个问题就找出了答案。如果琼无法解答最后两个问题，基拉就需要继续向其他参与者了解情况。

如果按照上述步骤开展对话，就一定能获得自己需要的所有信息。谨记，后续问题是对话的关键部分。必须合理安排问题，激励参与者给出需要的回答。

附加信息要求

讨论的下一主题是附加信息要求。这里的目标是确定用户是否需要当前未提供的其他信息。如果事实确实如此，就必须确认他们所需的附加信息类型，然后定义新数据结构，在后续的设计过程中支持这些附加信息。

引导参与者回顾他们当前收到的报表，开启对话。询问他们是否希望在报表中看到其他信息。接下来，将话题转向附加信息、该信息将影响到的报表，以及他们认为该信息必要的原因。然后，判断附加信息是否表示新主题或新特征。如果是，则对每个新项一一确认，并添加到对应的列表中。最后，回顾参与者的评论并判断是否有需要进一步讨论的问题。下面的示例将展示这一过程。

假设你刚刚和参与者回顾了他们当前使用的报表样本，由此开始讨论。其中一位参与者回顾的是图 6.11 所示的报表。

Current Product Inventory				
Product ID	Product Description	Category	SRP	Quantity
9001	Shur-Lok U-Lock	Accessories	75.00	
9002	SpeedRite Cyclecomputer		65.00	20
9003	SteelHead Microshell Helmet	Accessories	36.00	33
9004	SureStop 133-MB Brakes	Components	23.50	16

图 6.11 一位参与者所回顾的报表

现在，让该参与者记下她想要在这份报表中看到的附加信息及其原因。实际上，只要她的附注清楚并且与报表有明显关联，其他一切都无关紧要。有鉴于此，她决定使用大的便利贴记录。她指出了两个新字段并附注了理由。还将它们的名称写在了报表上，以此暗示其可能位置。图 6.12 所示为该样本报表。

Product ID	Product Description	Vendor Name	Category	Wholesale Cost
9001	Shur-Lok D		Accesso	
9002	S		Access	
9003	S		Access	
9004	Sur		Componen	33
				16

Can we include the Vendor name? It would make it easier to identify a specific product

If we could see wholesale cost, it would help us calculate more accurate discounts.

图 6.12 附注参与者评论的报表样本

接下来，确定附加信息中是否有新主题或新特征。运用确定主题技巧和确定特征技巧鉴定报表附注的评论。下列为图 6.12 中的经鉴定后的第一条评论：

“我们能否将供应商名称包含进去吗？这样就能轻易识别特定产品了。”

在此，同时找出了一个主题和一个特征。（注意，该主题和特征并无直接关联：“供应商名称”是供应商的特征，而与产品无关。不过，应该意识到这种主题和特征混搭的情况是常见的。在后续的设计过程中，将要解决这一问题。）这样，就可以检查一下主题列表和特征列表，看看这两项是否已经包含其中。如果这两项早已添加到两个列表中，则继续鉴定下一个评论，之后再重复这一过程。

如果发现了一个新主题，就将它添加到主题列表中，然后围绕这一主题寻找特征。完成后，将这些新特征添加到特征列表中，继续下一步，重复整个程序。然而，在多数情况下，你找到的只有新特征。不必为此紧张，因为

人们往往寻思着为报表中已发现的主题找到新的特征。

最后,复查每份报表,看看对参与者所做的附注是否有疑问或不解之处。例如,某位参与者在给定报表上表示特定字段是必要的,你可能对此依据的理由持有疑问。或者另一位参与者希望从其报表中删除某字段,你也许想知道其中的原因。当然,你想确保他建议删除的字段的确是不需要的,这一行为不会对其他字段造成不利影响。不论哪种情况,字段的增减都会影响最终的数据库结构。

如果评注中仍留有隐患,则与相关参与者一起讨论,尽可能将它们解决掉。通常,向参与者咨询就能排除所有隐患。但是在某些情况下,有些问题需要到设计过程后期才能解决。例如,你可能注意到了某些字段出现在多份报表中。判断这些字段是否可以精简,一时之间难以决定,一直要等到开始定义字段和表结构时才能明确。当你遇到当前时段难以解决的问题时,先予以标记将之搁置一边,以待日后解答。

未来信息要求

最后一个讨论的主题是未来信息要求。目标是了解参与者眼中的机构未来发展必需的信息。一旦确认了这些信息要求,就能定义数据结构为这些信息提供必要支撑。

首先,需要确保每位参与者都对机构如何发展有一定的认识。机构发展的性质将决定参与者要求的新信息。如果有多人对这些问题都不了解,就需要从管理人员处获取这方面的信息,并将这些信息在讨论前向参与者传达。等到大家都熟悉这些问题,即可开始对话。

讨论开始时,先引导参与者思考机构的未来发展及其对他们日常工作的影响。通常,一些参与者难以就这一场景开展联想。如果发生这种情况,可以使用一些问题帮助他们集中思想,比如:

机构的发展将会对工作所需的信息量产生什么影响呢?

随着机构的发展,你认为对于自己有效履行职责还需要其他信息吗?

机构的发展会给你的日常工作量带来什么变化吗？

你能预测一下，随着机构的发展，工作中需要什么类型（即范畴，并不指具体的项）的新信息吗？

如果你的工作量随着机构的发展而上升，你认为是否需要新的信息呢？

记住，大多数参与者的答案只是一种猜测。他们也无法准确预测信息的类型。然而，如果你预先了解他们预想的信息要求，就能为此定义必要的数据结构，提前做好准备。

针对参与者的回答，应使用确定主题技巧找出新的主题，并将它们添加到主题列表中。然后，使用确定特征技巧找出与已有或新主题相关的细节，并将这些特征添加到特征列表中。

你可以根据设想描绘出新型报表或数据输入表单的草图，让参与者直观地了解他们未来需要的信息类型。这样，草图就能帮助你确定数据库需要的新主题或特征。如果为样本报表创建了多份草图，就要确保将它们分别存入不同的文件夹，文件夹附注清晰的标记。每个修订版本都应编码，以便与之前的修订进行比较。图 6.13 所示示例为未来报表的初步设计。

<i>1st Quarter Customer Sales Statistics</i>				
<i>Customer ID</i>	<i>Customer Name</i>	<i>Maximum</i>	<i>Sales Amounts</i>	
			<i>Minimum</i>	<i>vegage</i>
9001	Stewart Jameson	265.00	23.00	55.00
9002	Shannon Black	550.00	125.00	70.00
9003	Estela Rosales	250.00	35.00	36.00
9004	Timothy Ennis	325.00	20.00	25.00

图 6.13 新报表设计示例

与用户继续讨论，直到对了解到的未来信息要求满意为止。讨论完之后，就可以准备对管理人员的访谈了。

❖注意：本节中学到的所有技巧同样适用于管理人员访谈。因此，下一节将简要描述。

管理人员访谈

访谈过程的第二部分是管理人员访谈。这一轮访谈主要集中在以下问题上：

1. 管理人员当前接收到的信息类型。
2. 管理人员需要的附加信息类型。
3. 管理人员预计未来所需的信息类型。
4. 管理人员对机构总体信息要求的认识。

❖注意：下文中，将使用“管理人员”一词代指操控和指挥机构的人员。

评审当前信息要求

访谈第一阶段的目标是了解管理人员接收的日常信息类型，并确认是否与报表样本有差异。

访谈开始后，了解每位参与者的工作和与之对应的职责。管理人员通常对访谈感到千头万绪，不知从何说起，所以向其询问工作方面的事情能帮助他集中注意力。从其回答中，能了解到他如何使用报表上的信息以及他对该种信息的看法。

接下来，让每位参与者确认是否使用过你所收集的报表样本中的报表。如果某参与者反映未使用其中的任何报表，继续下一步；否则，拿出每份报表，让其帮助你检查是否有疏漏。如有必要，使用确定主题技巧。如果该管

理人员发现新的主题，则将之添加到主题列表中，再使用确定特征技巧找出该主题的特征。然后，将新特征添加到特征列表中。对每个样本报表重复这一过程。

继续向每位参与者确认是否收到与报表样本中不符的报表。如有收到，则收集新报表的样本，并与参与者一起评审。使用确定主题技巧和确定特征技巧找出报表中的主题（及其特征），然后将主题和特征分别添加到相应的列表中。最后，为该报表标注描述，并将之添加到报表样本集合中。不断重复这一过程，直到找出每一份新报表。

评审附加信息要求

讨论的下一个主题是管理人员的附加信息需求。其目标是了解是否需要当前报表所缺的附加信息。如果确实需要，就必须逐一确认所缺附加信息。在后续的设计过程中，要酌情定义新的数据结构来支持这些信息。如果管理人员不需要附加信息，就可以进入访谈的下一步了。

这里的讨论采用与用户访谈期间所用相同的技巧。其步骤如下：

1. 与参与者再次评审报表样本，并询问是否还需要添加到报表中的附加信息。
2. 让参与者将附加信息附注在相应报表上，指明该信息必要的原因。记住，附注不论形式，但应意思清楚明确，对号入座。
3. 找出附加信息中的新主题及特征，并添加到相应的列表中。
4. 回顾报表，如有任何疑虑，即与参与者进行讨论。解决掉这些问题，这一过程也就完成了。

评审未来信息要求

未来信息要求是接下来要讨论的主题。其目标是向管理人员了解预计未来需要的信息。待这些要求确认之后，就能确保有相应的数据结构支持这些信息。

讨论开始时，让参与者先思考一下机构当前发展方向。然后向他们了解

机构发展将对他们决策所需信息以及领导机构方式的影响。记住，他们的回答是基于猜测，你所提出的问题也基于这一目的；管理人员的预测难以确保准确，一切只能等到事情发生后才能揭晓。（不过，针对未来制订相应的计划始终都是有利的。）使用确定主题技巧和确定特征技巧从参与者的回答中找出新的主题和特征，并将它们分别添加到对应的列表中。

接下来，简要记下参与者想到的任何新报表。找出每份报表中的新主题和特征，添加到对应列表中。然后，将这些新报表存入带有明确标记的文件夹，文件夹则归入样本集合中。

待对管理人员预测的未来信息要求做了充分了解之后，就可以进入最后一个主题了。

评审总体信息要求

讨论的最后一个主题是机构的总体信息要求。在管理人员眼中，机构需要什么类型的信息呢？这里的目标是寻找用户访谈和管理人员访谈中未讨论到的、机构需要维护的数据。如果发现确实有这种数据，就必须为其设计相应的数据库结构。

取出分析和采访过程中收集到的所有报表，与参与者再次评审。然后让参与者考虑一下报表提供的信息以及他们如何使用这些信息。（注意，他们必须假设自己将如何使用新报表中的信息。）接下来，向参与者了解是否存在对机构有价值的信息尚未被发掘出来。如果他们确实存在此类信息，则针对这些新信息重复确定主题和确定特征的过程。获取对应信息的新报表样本，并将样本存入已有的新报表集合中。

例如，某位参与者确认了人口统计信息的需求；他坚信这一信息将帮助产品找到更为明确的目标市场定位。现有的报表中并不具备这一信息，于是你和他一起制作一个包含该信息的报表初稿以做确认。（实际上，他可能已有过多次这样的尝试，但这并不构成负面影响。）然后，使用相应技巧确定并标记报表中的主题和特征，并将该报表归入新报表集合当中。在后续的设计过程中，还要为这一新信息定义必要的数据结构。

不断重复这一程序，直到参与者找不出其他对机构有价值的信息。待确信明确了所有机构信息要求，先暂停访谈，开始编辑初始字段列表。

重要的是，必须明白这一过程也许还要重复，尽管你和参与者都认为已经找到所有机构可能用到的信息。随着设计过程的展开，通常会发现新的信息。

编辑完整字段列表

初始字段列表

既然当前数据库分析与用户和管理人员访谈已经完成，就可以创建一个初始字段列表。这个列表代表了机构基本数据要求，它包含了数据库定义的核心字段集合。创建初始字段列表包括两个步骤。

第一步：评审和精简特征列表

第一步是评审并精简分析和访谈过程中收集到的特征列表。第3章中已经提到，字段表示特定主题的特征；因此，特征列表中每一项都将成为一个字段。不过在此之前，先要评审特征列表，删除重复特征。

访谈期间，从参与者的回答中找出各种特征，并将它们归入一个列表中。或许，你会将相同的特征重复添加多次，或者无意中以多个不同的名称代指相同的特征。因此，需要对特征列表进行精简。

精简名称相同的项

首先，找出名称相同的项。当发现同一名称出现多次时，先确认它们是否表示同一特征。如果表示同一特征，则删除多余项保留其中一个；否则，逐一确认每个名称所指。通常，你会发现某一重复名称表示的是不同主题的相似特征。对于这种情况，可以重新命名，使其与对应主题相符。

例如，“名称”一项在特征列表中出现了三次。起初的想法可能是删除其中两项，因为当前的目标是消除重复特征。然而在此之前，应该先判断每一项是否表示同一特征。通过检查访谈记录轻易就能得出结论；它能帮助你

记起添加该项的时间和原因。

仔细检查之后，发现第一个“名称”表示“客户”的特征，第二个表示“员工”的特征，第三个表示“联系人”的特征。可以为之重新命名（使用主题作为前缀），反映其真实含义。这样，三个新特征分别为“客户名称”、“员工名称”，以及“联系人名称”。

在特征类别中，类似情况比较常见。因此，必须以相同的方式解决。通常，多次出现的项包括“地址”、“城市”、“国家”、“邮政编码”、“电话号码”，以及“电子邮箱”。你可以将它们统称为**通用项目**。关键在于为通用项目的每一个示例重新命名，表现出它与对应主题的真实关系，尽可能确保字段列表的准确性。

精简表示相同特征的项

寻找表示**相同特征**的项，删除多余的项，保留其中一个。目标是保证特征列表中所有特征仅出现一次。例如，特征列表中有“产品#”、“产品 No.”和“产品编号”三项。显然，这些项表示同一特征，列表中只需保留一项。选取意思最为明确完整的一项保留下来，清除多余的其他项。（在这个案例中，最好的选择是“产品编号”，因为它符合以上标准。）

确保项与特征正确对应

最后，确保列表中每一项表示一个**特征**。通常容易将表示主题的项混入特征列表当中。通过下列问题，可以对每一项进行测试：

这个词能用来描述事物吗？

这个词是否表示事物的成分、细节或组成部分？

这个词是否表示一个集合？

这个词所指的事物是否可以进行细分？

问题的复杂程度取决于你所面对的项。如果发现某一项表示主题而非特征，则将该项删除并添加到主题列表当中。找出新主题的特征，并添加到特

征列表中。

例如，特征列表中出现了“Item（项目，商品）”，你并不能确定它表示特征还是主题。可以借用下列问题进行判断。

“Item”可以用来描述事物吗？

“Item”是否表示事物的成分、细节或组成部分？

你会认为“Item”可以描述一单销售，因为它表示顾客购买的商品。另一方面，你也可以说“Item”不是特征，因为它不表示销售的单一方面。比如，“出售日期”表示销售的单一特征。先将疑问放到一边，再来看下面的问题：

“Item”表示一个集合吗？

想想它的复数形式“Items”，这个问题就比较容易了。如果“Items”表示一个集合，它就是一个主题。问题开始明朗起来，“Item”确实表示一个集合。再看下面的问题，你就可以下结论了。

“Items”所指的事物是否可以进行细分？

看看是否能找出“Items”的特征，这个问题就迎刃而解了。如果能够找出它的特征，那么“Items”就一定表示主题，应该将它归入主题列表中。你也应该找出其特征，并将它们添加到特征列表中。

继续这个过程，直到评审和精简工作达到满意的效果。完成之后，就有了初始字段列表的第一个版本。下一步，你将添加新的项，并且做进一步的精简。

第二步：确认样本中是否有新的特征

这一步是检查分析过程中收集到的所有样本。目标是确认样本中是否有新特征应该添加到初始字段列表中。

首先，聚焦样本中发现的每一个特征。然后，检查每一个特征，确认是否已包含在初始字段列表中；如果出现，则在样本中划掉该项。接着，研究剩下的特征，确认其中是否有与现有字段意义相同的项；如果有，则在

样本中划掉该项。(使用第一步中的方法做出判断。)最后,将样本中剩余的特征添加到初始字段列表当中。

例如,你使用的是图 6.14 所示的数据收集样本。

The form is titled "Mike's Bike Shop" and has a menu bar with File, Edit, View, Insert, Format, Records, Tools, Window, and Help. Below the menu bar is a toolbar with icons for file operations. The form is divided into two main sections: "Supplier Information" and "Contacts".

Supplier Information:

Company:	Acme Power Tools		
Address:	635 Montana Ave	Status:	Active
City:	El Paso	Office Phone:	598-4455
State:	TX	Zip:	79925
		Website:	www.apr.com

Contacts:

Name:	George Quinn	Phone No.:	532-9228
Name:		Phone No.:	

At the bottom of the form are four buttons: "< Back", "Next >", "Edit", and "Save".

图 6.14 数据收集样本示例

聚焦图 6.15 所示样本中发现的每一个特征。

This figure shows the same form as Figure 6.14, but with some text highlighted in yellow to indicate specific features. The highlighted text includes:

- Company: Acme Power Tools
- Address: 635 Montana Ave
- City: El Paso
- State: TX
- Zip: 79925
- Status: Active
- Office Phone: 598-4455
- Website: www.apr.com
- Name: George Quinn
- Phone No.: 532-9228

The rest of the form, including the menu bar, toolbar, and buttons, remains the same as in Figure 6.14.

图 6.15 突出特征的样本

你可能会发现某些样本中多个特征出现了多次。如你所见,“Name”和“Phone No.”在这个样本中都出现了两次。可以将多余的划掉,因为它们意

思相同。

继续讨论这个例子，比如评审初始字段列表，发现样本中每个特征都已经包含在列表里，只有“Name”和“Phone No.”除外。不过，在将“Name”和“Phone No.”添加到初始字段列表中前，先确保这些项的名称准确描述了与主题的关系。在这种情况下，剩下的两项表示“Contacts（联系人）”的特征。因此，将这些特征重命名为“Contact Name”和“Contact Phone Number”，然后再添加到初始字段列表中。针对收集的每个样本重复这一过程，直到检查完所有样本。这样，就有了初始字段列表的第二个版本。

附注：值列表

检查数据库、电子数据表格或网页样本中的特征时，记录下每个包含值列表（也称为枚举列表）的特征名称。这个列表可以为相应特征指定可接受的值范围，它往往强制施行某一给定的业务规则。（业务规则详情见第 11 章“业务规则”。）例如，你供职于一家制造企业，该企业有四个供应商负责将商品销售给全国的顾客。你可以使用一个值列表，确保用户选择其中一个供应商为其运输货物。图 6.16 展示了这个例子（注意“Ship Via”），同时还展示了两组常见的值列表。

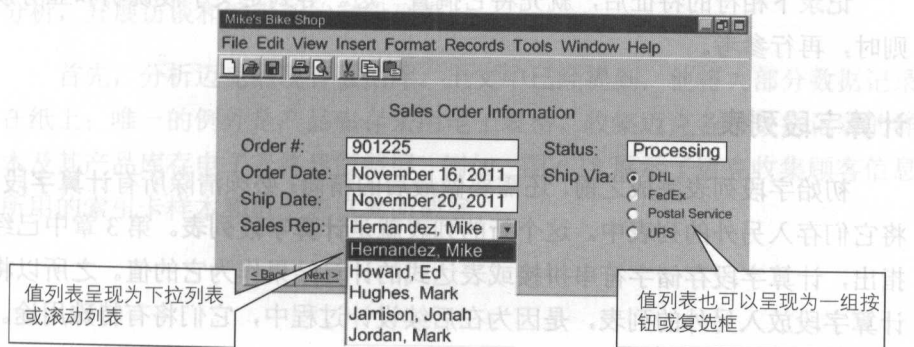


图 6.16 数据库界面（包含两个值列表）

当记录包含值列表的特征名称时，也要记录下列表中的值。如果列表包含的值很多，则对值的类型进行简单描述，如有可能，还应介绍最大值和最小值；否则，记录下每一个值。图 6.17 所示为记录示例。

Characteristics Incorporating a Value List	
Characteristic	Value List
Category	Accessories, Bikes, Clothing, Components, Maintenance, Racks, Wheels
Department	Accessories, Bikes, Clothing, Service,
Sales Rep	The name of every employee within the organization whose position is that of sales rep
Ship Via	DHL, FedEx, Postal Service, UPS

图 6.17 包含值列表的特征

可以对需要记录的特征进行一番甄别。比如，无须记录包含“是/否”、“对/错”或“活动/不活动”这类简单值的特征。反之，取值较为特殊、具体的特征则应予以记录。

记录下相符的特征后，就先将它搁置一边。等到定义字段说明和业务规则时，再行参考。

计算字段列表

初始字段列表完成之前，还需要做最后的精简：必须清除所有**计算字段**，将它们存入另外的列表中。这个新的列表就是**计算字段列表**。第3章中已经指出，计算字段存储字符串拼接或表达式的计算结果作为它的值。之所以将计算字段放入另外的列表，是因为在后续设计过程中，它们将有特殊用途。

使用初始字段列表中的已有字段建立计算字段列表。检查初始字段列表，确认是否有符合计算字段描述的字段。如果字段名称中包含特定的词，如**数额**、**总计**、**合计**、**平均**、**最大值**、**最小值**以及**数量**，则该字段可能为计算字段。计算字段通常有“小计”、“平均年龄”、“折扣金额”，以及“客户

数量”。待计算字段确认之后,即将该字段从初始字段列表中移除,并添加到计算字段中。检查完初始字段列表后,就有了两个新的列表:初始字段列表第三版和计算字段列表。

与用户和管理人员评审列表

对用户和管理人员开展简单的访谈,和他们一起评审初始字段列表和计算字段列表。目标是确认是否两个列表有疏漏之处。如果大家一致对两个列表表示满意,就可以进入下一步;否则,找出遗漏的字段,并将该字段添加到相应的列表当中。访谈结束之后,就将获得这两个列表的最后版本。

确保访谈如期开展,因为参与者的反馈能帮你验证列表中的字段。不妨再提醒一遍,切勿过早认为这些列表已经非常完善。即便到了这一步骤,可能仍然未能将所有必要的字段全部收集起来,或许不经意间就遗漏了一些字段。不过,如果认真仔细地制作列表,微小的疏忽也是比较容易解决的。

案例分析

先前已经为迈克的新数据库定义了宗旨和任务目标。现在,就应该展开分析,开展访谈和编辑初始字段列表。

首先,分析迈克的现有数据库。上文中已经提到,他将大部分数据记录在纸上;唯一的例外是产品库存采用电子表格。收集迈克各种数据记录的样本及其产品库存电子表格界面截屏。例如,图 6.18 展示了迈克收集顾客信息所用的索引卡样本和电子表格界面截屏。

<i>Steven Horst</i>	<i>363-9755</i>
<i>Apartment 2B</i>	
<i>2380 Redbird Lane</i>	
<i>Seattle, WA 98115</i>	
<i>He's primarily interested in mountain bike stuff.</i>	
<i>Keep him abreast of the summer bike tours.</i>	

Mike's Bike Shop - Product Information					
File Edit View Insert Form at Tools Data Window Help					
	A	B	C	D	E
1	Product ID	Product Description	Category	SRP	Qty On Hand
2	9001	Shur-Lok U-Lok	Accessories	75.00	
3	9002	SpeedRite Cyclecomputer		65.00	20
4	9003	SteelHead Microshell Helmet	Accessories	36.00	33
5	9004	SureStop 133-MB Brakes	Components	23.50	16
6	9005	Diablo ATM Mountain Bike	Bikes	1,200.00	
7	9006	UltraVision Helmet MountMirrors		7.45	10

图 6.18 迈克自行车行纸质和电子表格样本

接下来，确认迈克呈现信息所使用的方法。当前，他和他的员工利用了各种各样的报表展示信息，并根据这些信息开展日常事务。生成大多数报表所使用的工具是文字处理程序。收集所有报表的样本，将它们存入一个文件夹中，留待后用。图 6.19 所示为迈克在计算机上创建的一个报表样本。

Supplier Phone List		
Company Name	Contact Name	Phone Number
ACMECycle Supplies	George Chavez	633-9910
B & M BikeSupplies	Carol Ortner	527-3817
Cycle Works	Julia Black	527-0019
Evanstone'sCycle Warehouse	Allan Davis	636-9360

图 6.19 一份迈克自行车行的报表样本

现在，可以采访迈克的员工了。开展访谈时，应记住以下几点。

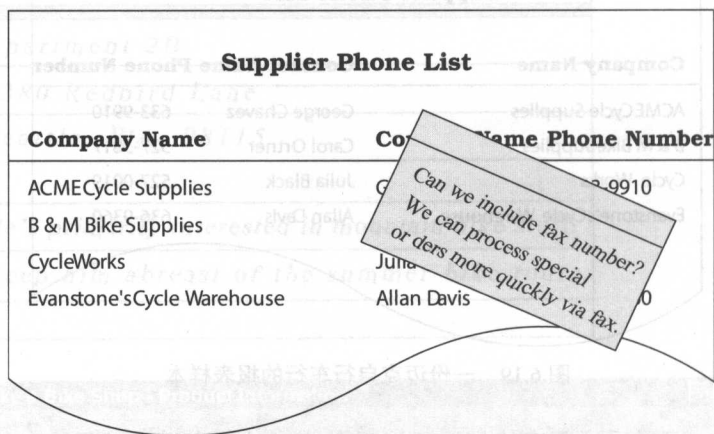
- 了解员工使用的数据类型及其使用方式。使用确定主题技巧和确定特征技巧分析他们所做的回答以及制订后续问题。
- 评审分析开始阶段收集的所有样本。确认每个样本的使用方式，写下适当的描述，并将描述附注于该样本上。
- 了解员工的信息要求。了解他们当前使用的信息类型、所需的附加信息类型（附上样本），以及他们预计未来需要的信息类型。

假如访谈期间，某位员工想知道能否将一个新字段添加到供应商电话列表报表中。你将如何回答呢？可以将报表给他，让他附注新字段名称及简要的理由。之后，再将该样本放回到报表样本文件夹中。图 6.20 所示为加上附注的报表样本。

与迈克进行最后访谈。访谈时，记住以下几点。

- 了解迈克当前收到的报表；必须清楚他制定业务决策所依据的信息类型。如果报表样本文件夹中没有这些报表，则收集每份报表的样本，并将样本添加到文件夹中，并根据需要更新主题和特征。
- 和他一起评审报表样本，检查是否有疏漏的主题或特征。使用适当的技巧确定这些主题和特征，并将它们添加到相应列表中。
- 向他确认是否有需要补充的附加信息。

- 让他预计未来需要的信息类型。

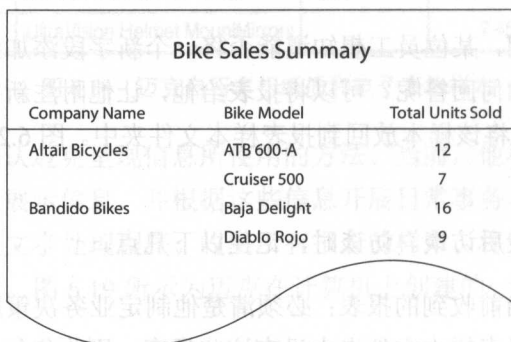


Company Name	Company Name	Phone Number
ACMECycle Supplies		9910
B & M Bike Supplies		
CycleWorks		
Evanstone'sCycle Warehouse	Allan Davis	

Can we include fax number?
We can process special
orders more quickly via fax.

图 6.20 加上附注的报表样本

在和迈克讨论他的未来信息需要时，他表示一旦开展业务，想要接收到的一种信息是生产商自行车销售总量。他认为，这种信息能帮助他确定应该长期供应的自行车种类。鉴于当前并没有这样的报表，应该让迈克制作一个初样。然后，找出该报表中的主题和特征，并将它们添加到相应的列表中。之后，再将该报表归入报表样本文件夹中。图 6.21 所示为迈克的新报表草图。



Company Name	Bike Model	Total Units Sold
Altair Bicycles	ATB 600-A	12
	Cruiser 500	7
Bandido Bikes	Baja Delight	16
	Diablo Rojo	9

图 6.21 迈克新报表的草图

至此，分析完毕。通过访谈收集到了相关的所有样本，分别创建了一个主题列表和特征列表。部分主题列表和特征列表见图 6.22。接下来，创建初

始字段列表。

List of Subjects <i>as of 02/13/12</i>		List of Characteristics <i>as of 02/13/12</i>	
Customers	Sales	Address	Home Phone
Employees	Suppliers	Birth Date	Last Name
Products		Category	Name
		City	Phone
		Comments	Product No.
		First Name	State

图 6.22 主题列表和特征列表的部分

前面已经提到，建立初始字段列表的第一版首先要精简特征列表。清除所有重复特征以及表示相同特征的项，精简采用通用名称的项。（还记得前面提到的“名称”这一特征吗？如果发现此类特征，就使用相应办法解决。）接下来，评审所有样本，确认其中是否有初始字段列表中未出现的特征。将相应特征添加到列表中。完成这些步骤后，就有了初始字段列表的第一版。

然后，从初始字段列表中移除所有计算字段，并将它们归入计算字段列表；这样，就得到了新的计算字段列表。图 6.23 所示为部分最终初始字段列表和计算字段列表。

Preliminary Field List <i>as of 02/16/12</i>		Calculated Field List <i>as of 02/16/12</i>	
Birth Date	Office Phone	Discount Amount	
Employee Name	Product Name	Grand Total	
Employee Address	Category	Item Total	
Employee City	Unit Price	Subtotal	
Customer Name	Invoice Number		
Customer Address	Invoice Date		

图 6.23 初始字段列表和计算字段列表的部分

❖注意：读者可能已经注意到，每个列表的标题都包含了日期。使用日期标明制作列表时间，从而清晰记录这一制作过程。

小结

本章开头即讨论了分析机构现有数据库的原因。分析有助于了解现有数据库的各个方面，对设计新的数据库有重要作用。通过分析了解到的信息，就可以设计出符合机构要求的数据库。接着，又简单探讨了机构常用的两种数据库：纸质数据库和遗留数据库。结束时，介绍了分析过程的三个步骤：评审收集数据的方式，评审信息呈现的方式，以及开展机构员工访谈。

然后就是评审环节。这一部分介绍了如何评审机构收集数据的方式和如何获取数据收集样本集合。另外，还叙述了如何评审机构呈现信息的方式及如何获取报表样本集合。

接下来，我们讨论了访谈，解释了这一阶段开展访谈的原因。讨论中提到了访谈成功的两项关键技巧：确定主题技巧和确定特征技巧。

随后讨论的主题是开展用户访谈。谈到了访谈中必须解决的四个问题，连带介绍了解决这些问题的技巧。讨论管理人员访谈时，也提到了相似的内容。

最后是基于特征列表和样本中出现的特征编辑字段列表的过程。将字段列表分解为初始字段列表和计算字段列表。初始字段列表代表着机构基本数据要求，包含了数据库中必须定义的核心字段。计算字段则是指包含字符串连接或表达式计算所得值的字段。

思考题

1. 指出分析现有数据库的两个目标。
2. 判断题：现有数据库结构可以作为新数据库结构的基础。

3. 什么是遗留数据库?
4. 说出分析过程的两个步骤。
5. 分析过程中应该评审哪些类型的计算机软件程序?
6. 得到数据收集和信息呈现样本后,为什么要开展访谈?
7. “开放式”和“封闭式”问题分别如何使用?
8. 什么是确定主题技巧?
9. 如何为特定主题确定特征?
10. 判断题:用户和管理人员应同时进行访谈。
11. 必须确认哪三种基本信息要求?
12. 什么是初始字段列表?
13. 说说为什么初始字段列表中的每一项都应该有独特的名称。
14. 什么是值列表?
15. 什么是计算字段?你应该为之做些什么?

本章主要介绍数据库设计的基本概念、数据库设计的方法和步骤、数据库设计的评价标准、数据库设计的应用案例等。



第7章

建立表结构

未得到数据就妄加揣测是一个严重的错误。

——夏洛克·福尔摩斯

《福尔摩斯冒险史》

本章内容

定义初始表列表

定义最终表列表

字段对应入表

精简字段

精简表结构

案例分析

小结

思考题

各种机构使用数据库记录各个重要主题。例如，医疗诊所记录患者、医生以及预约等；设备租赁企业必须维持顾客、设备以及租赁合同的相关数据；教务办公室负责的方面（至少）包括学生、教职员工以及课程。无论上述哪种情况，或者你可以想到的任何情况，数据库中的每个表都代表一个主题。此外，每个表都由字段组成，字段表示对表主题进行定义和描述的特征。表

构成了数据库的基础。设计得当，就能确保基础稳固。

定义初始表列表

这一阶段的任务是定义初始表列表，为新数据库确定和创建表做准备。建立这个列表包括三个步骤。第一步涉及使用初始字段列表，第二步是使用访谈过程中收集到的主题列表，第三步是使用数据库设计过程之初定义的任务目标。之后，将利用初始字段列表中的字段建立每个表的结构。

确定隐含主题

定义表首先要评审初始字段列表。目标是鉴定列表中字段隐含的主题。

读者也许会疑惑，为什么是评审初始字段列表，而不是使用主题列表。主题列表似乎更为直截了当。毕竟，这个列表是在访谈过程中精心建立起来的。而且，你从用户和管理人员的交流中收获颇多。无疑，所有这一切帮助你找到数据库所需的每一个主题。不过，如果存在失误，就会造成表的缺失。

分析初始字段列表中的字段，让你从公正的角度发现主题——让字段和你“说话”。关键是抱着尽量客观的心态审视列表，最好要做到视若初见，不帶有任何访谈中形成的成见。这样，就能看清哪些字段揭示哪个主题，有些可能是访谈期间未能发现的。你也能使用初始字段列表验证主题列表中的主题。通过这样的方法，就能交叉验证之前的成果，确保新数据库包含所有必要的主题。

评审初始字段列表时，要考虑特定字段组是否围绕某一主题。待确定无误后，继续下一组字段。如果从字段中得出一个主题，则将该主题输入新的初始表列表中。图 7.1 所示为部分初始字段列表，同时也展示了如何从一组字段中确定主题。

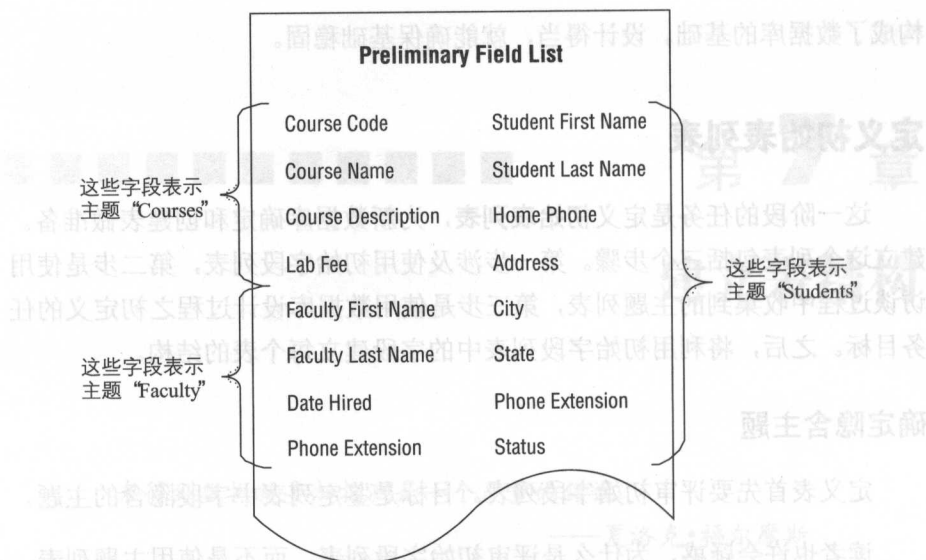


图 7.1 使用初始字段列表确定主题

将所有字段评审完，尽可能找出更多的主题。确保所有主题都加入了初始表列表中。在接下来的步骤中，这个列表还将继续扩充。图 7.2 所示为第一版初始表列表示例。

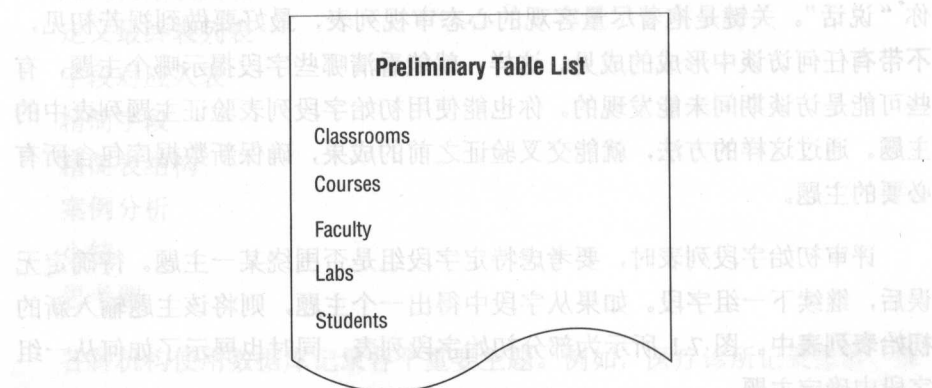


图 7.2 第一版初始表列表

使用主题列表

将主题列表（用户和管理人员访谈期间所建立）和第一版初始表列表合

并，建立第二版初始表列表。这一新版本包含了更为完整的表列表。合并两个列表分为三个步骤，包括消除重复项，消除表示相同主题的项，以及将剩余项合并为一个列表。

第一步：消除重复项

首先，评审和对照检验主题列表与初始表列表中每一项。目标是找出**重复项**，即同时出现在主题列表与初始列表中的项。必须谨慎对待发现的重复项。虽然名称相同，但是要先确认它们是否表示**不同的主题**。（必要时，使用访谈笔记进行查证。）如果它们确实表示不同主题，为每一项重新命名，保证它们准确表达所代表的主题，然后将它们都添加到初始表列表中；否则，判断它们是否表示同一主题。如果断定两项表示同一主题，将主题列表上的该项划掉，保留初始表列表中的一项。继续审查，直至完成两列表中所有项。现在，我们不妨来示范一下。

假设，你正在为一家设备租赁公司开发数据库，使用的是图 7.3 所示的主题列表和初始表列表。

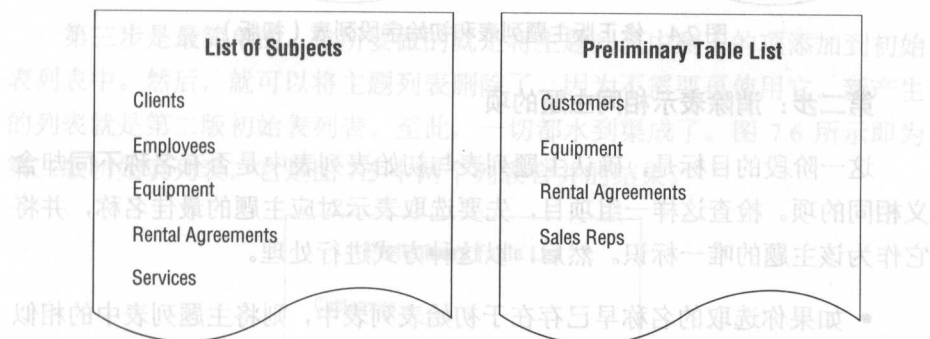


图 7.3 某设备租赁公司的主题列表和初始表列表

评审这些列表，会发现两对重复的项：“Equipment（设备）”和“Rental Agreements（租赁合同）”。这些项有待进一步的验证，所以先从“Equipment（设备）”着手，设法确认两项是否表示不同主题。回顾访谈记录，发现主题列表中“Equipment”表示诸如工具、电器和视听设备的设施。你又记起初始表列表中的“Equipment”也包含卡车、货车和拖车。进一步检查访谈记录，

得知交通工具的租赁有别于“常规”设备的租赁。因此，两个“Equipment”确实表示不同的主题。保留其中一个，并将另一个改为“Vehicles(交通工具)”，这样就消除了重复项。然后，将这两项都添加到初始表列表中。

现在，再来看看“Rental Agreements”。幸运的是，你发现两项表示完全相同的意思。这样，唯一要做的就是划掉主题列表中的“Rental Agreements”。接下来，继续检查，直至完成主题列表中所有的项。图 7.4 所示为修正过的主题列表和初始表列表。

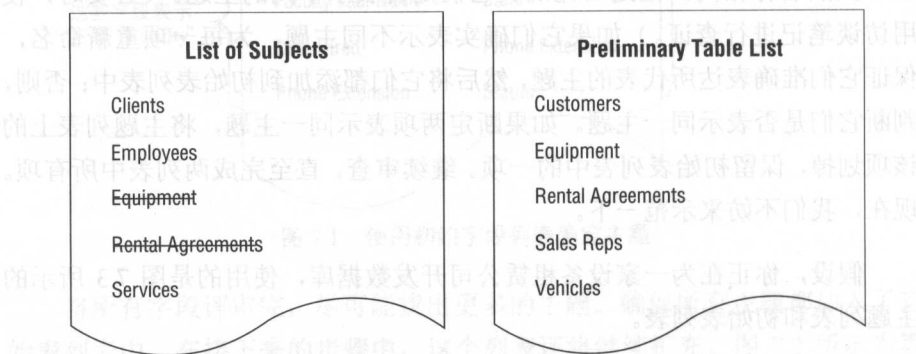


图 7.4 修正版主题列表和初始字段列表（初版）

第二步：消除表示相同主题的项

这一阶段的目标是，确认主题列表与初始表列表中是否有名称不同却含义相同的项。检查这样一组项目，先要选取表示对应主题的最佳名称，并将它作为该主题的唯一标识。然后，以这种方式进行处理。

- 如果你选取的名称早已存在于初始表列表中，则将主题列表中的相似项划掉。
- 如果该名称出现在主题列表中，则用它代替初始表列表中的相似项。

重复这一过程，直至检查完主题列表中的每一项。

继续设备租赁公司的范例，假设你发现主题列表中“Clients”与“Employees”和初始表列表中“Customers”与“Sales Reps”分别表示相同的主题（参见图 7.4）。你决定先解决“Clients”和“Customers”，回顾访谈

记录, 确定“Customers”这一名称用于表示向该公司租赁设备的个人或组织最恰当。然后, 将“Clients”划掉, 保留“Customers”。接着是下一对, 你认为应该保留“Employees”, 清除“Sales Reps”, 因为“Employees”不牵涉职务, 用来描述该公司雇佣的人员更为合适。图 7.5 所示为修正后的列表, 其中也包含对重复项的处理。

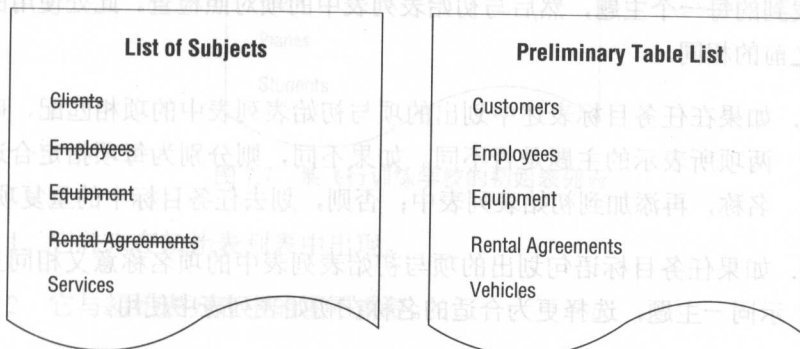


图 7.5 修正版主题列表和初始表列表 (第二版)

第三步: 合并主题列表和初始字段列表中的项

第三步是最简单的。你所要做的就是将主题列表中剩下的项添加到初始表列表中。然后, 就可以将主题列表删除了, 因为不需要再使用它。新产生的列表就是第二版初始表列表。至此, 一切都水到渠成了。图 7.6 所示即为第二版初始表列表, 它是图 7.5 中两个列表合并的结果。



图 7.6 第二版初始表列表

使用任务目标

在这个步骤中，将使用目标任务检验前两个步骤中是否遗漏主题。这是最后的机会。

从第一个任务目标着手，使用确定主题技巧找出其中暗含的主题。划出你所找到的每一个主题，然后与初始表列表中的项对照检查。此处使用的技巧与之前的相同。

1. 如果在任务目标表述中划出的项与初始表列表中的项相匹配，确认两项所表示的主题是否不同。如果不同，则分别为每项指定合适的名称，再添加到初始表列表中；否则，划去任务目标中的重复项。
2. 如果任务目标语句划出的项与初始表列表中的项名称意义相同且表示同一主题，选择更为合适的名称在初始表列表中使用。
3. 如果任务目标语句中出现了新的主题，则将之添加到初始表列表中。

重复以上步骤，直至完成所有任务目标。下面将举例说明如何使用这些技巧评审任务目标。

假设，你在为一家飞行训练学校设计数据库。你刚刚开始这个过程，对下列语句运用了确定主题技巧：

我们需要维护飞行员及他们的资质证书方面的数据。

现在，将上述任务目标中划出的主题与图 7.7 中初始表列表中的项进行对照检查。

在此，划去任务目标中的“飞行员”，是因为它早已出现在初始表列表中，且表示同一主题。然后，进一步审视“资质证书”，仔细考虑之后，你得出以下发现。

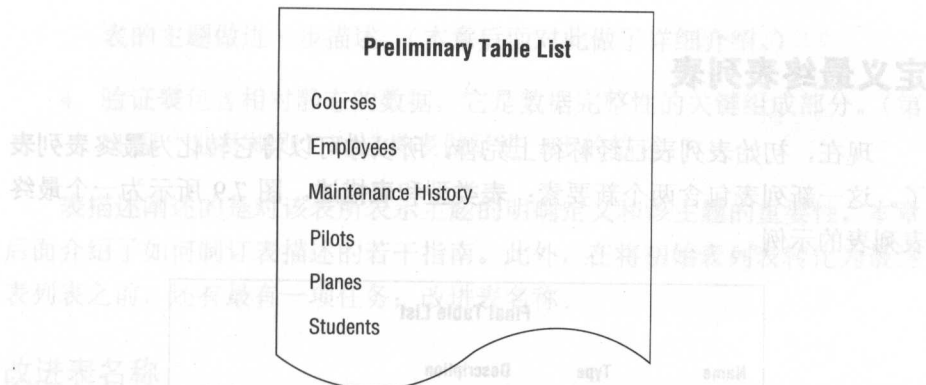


图 7.7 某飞行训练学校的初始表列表

1. 它并未在初始表列表中出现。
2. 它与初始表列表中的项不重复。
3. 初始表列表中没有与其名称意义相近的项。
4. 它所表示的主题与初始表列表中的项不重复。

以上发现表明“资质证书”表示新的主题，应添加到初始表列表中。所以将之添加到初始表列表中，再从目标任务中划去。图 7.8 展示了初始表列表的又一修正版。

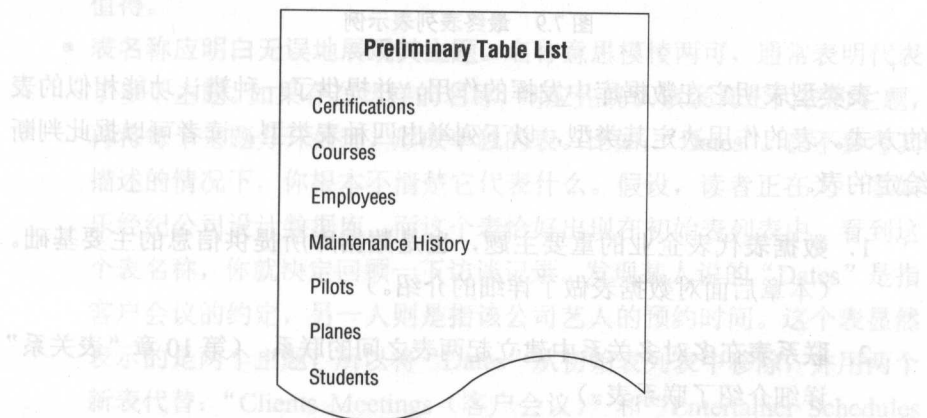


图 7.8 修正后的初始表列表

定义最终表列表

现在，初始表列表已经称得上完善，所以你可以将它转化为最终表列表了。这一新列表包含两个新要素：**表类型**和**表描述**。图 7.9 所示为一个最终表列表的示例。

Final Table List		
Name	Type	Description
Classrooms	Data	The spaces or areas within a facility reserved for the purpose of conducting class proceedings. Information regarding the physical aspects, on-site resources, and availability of these areas is useful because it allows us to assign classes to the facility that can make the best use of these areas.
Courses	Data	The programs of instruction conducted through courses offered by this institution. Course information must always reflect the addition of new courses, the deletion of old courses, and the continuing evolution of existing courses.

图 7.9 最终表列表示例

表类型表明它在数据库中发挥的作用，并提供了一种辨认功能相似的表的方式。表的作用决定其类型，以下列举出四种表类型，读者可以据此判断给定的表。

1. 数据表代表企业的重要主题，也是数据库所提供信息的主要基础。（本章后面对数据表做了详细的介绍。）
2. 联系表在多多关系建立起两表之间的联系。（第 10 章“表关系”详细介绍了联系表。）
3. 子集表包含与特定数据表相关的字段，以非常具体的方式对该数据

表的主题做进一步描述。(本章后面对此做了详细介绍。)

4. 验证表包含相对静态的数据,它是数据完整性的关键组成部分。(第11章“业务规则”对这类表做了进一步的描述。)

表描述阐述的是对该表所表示主题的明确定义和该主题的重要性。本章后面介绍了如何制订表描述的若干指南。此外,在将初始表列表转化为最终表列表之前,还有最有一项任务:改进表名称。

改进表名称

制定表名称也许比想象的要更难。在第3章“术语”中已经提到,一个表只表示一个主题;因此,其名称必须清晰地表明其所代表的主题。下列指南旨在帮助你制定简明扼要的表名称,同时保证名称前后一致。

制订表名称指南

- 制订的表名称应独特且有意义。采用唯一的名称,确保每个表表示不同的主题,确保企业内部人员都理解其含义。(如果出现重复的名称,则使用前面提到的技巧进行处理。)选取有意义的名称,达到不言自明的效果。“汽车维修”就是一个很好的例子,这个名称让人一目了然。制订独特且到位的名称确实需要花费心思,但从长远来看,确实值得。
- 表名称应明白无误地展现其主题。名称意思模棱两可,通常表明代表了多个主题。如果存在这样的名称,则应先确认该表真正代表的主题,再将每个主题分开处理,形成单独的表。比如,“Dates”。在不参考其描述的情况下,你根本不清楚它代表什么。假设,读者正在为一家娱乐经纪公司设计数据库,而这个表恰好出现在初始表列表中。看到这个表名称,你就决定回顾一下访谈记录。发现某人说的“Dates”是指客户会议的约定,另一人则是指该公司艺人的预约时间。这个表显然表示的是两个主题,所以将“Dates”从初始表列表中移除,并用两个新表代替:“Clients Meetings (客户会议)”和“Entertainer Schedules (艺人时间表)”。

大概最为模糊的表名称是“杂项”，难以言明其具体所指。可能偶尔也会不得已用到它，因为无法明确说明初始字段列表的某些字段。如果出现这样的情况，那就不妨暂停下来，休息片刻，然后再来看看，重新检查这些字段。应用所学到的设计技巧，仔细且有条不紊地对待它们，一定会知道究竟应该如何处理。

- **表名称应尽量精简。**企业内部人员应能直接理解其含义，而不必借助描述。虽然目标是要制订简洁的名称，但应避免过于简约。“TD-1”就是一个很好的反例。除非清楚名称中每个字母的含义，否则根本无从得知这个表的主题。当然，也不能太过烦琐。“多用途汽车维修设备”这个名称就显得过于冗长，可精简为“设备”。
- **避免使用描述物理特征的词语。**表名称中应避免使用诸如文件、记录和表的词语，因为这会让人产生徒增困惑。包含这类词语的表名称极有可能代表多个主题。考虑一下“患者记录”。表面上，这似乎是个不错的名称。然而，花些时间想想它所表达的含义，就会意识到有潜在的问题。这一名称包含三个主题：“患者”，“医生”和“检查”。想到这一点，应该将初始表列表中的“患者记录”移除，用三个新表代替，每一个都表示一个主题。
- **避免使用缩略语。**缩写词难以解释，简写词无法传达主题，两者都违背了第一条指南。就拿缩写词举个例子。比如，你在帮某家公司改进其数据库结构，碰到一个表名称“SC”。如果不清楚字母本身的含义，怎么能知道表所表示的主题呢？事实上，无法轻易确定表的主题。此外，可能还会发现该表对不同部门有不同的含义。于是，决定与一些员工展开访谈，以确定这些字母的意思。（现在，可怕的事情来了。）让人难以置信的是，你发现人事部的员工认为它表示“指导委员会（Steering Committees）”；信息系统的员工则觉得它指的是“系统配置”；安保部门的员工则坚持认为是“验证码”。这个例子清楚地说明了表名称中避免使用缩略语的原因。
- **避免使用专有名称或其他过多限制输入数据的词语。**这样就能避免掉入重复表结构的陷阱中。例如，“西南地区员工”这一名称严重限制了输入这个表中的数据。随着企业扩张，如何处理其他地区的员工

呢？当这家企业开始招聘华盛顿州、俄勒冈州和爱达荷州的人员时，就必须创建一个“太平洋西北地区员工”的表。等到有了亚利桑那州、犹他州、内华达州和加利福尼亚州这些地方的员工，又要建立“西部地区员工”的表。

根据正确的数据库设计指南，应避免创建这样的重复结构，因为这很麻烦。

1. 用户难以同时从三个表中检索数据。
2. 数据库维护人员有责任确保表的结构始终保持协调。如果维护人员添加、更改或删除某表中的一个字段，必须对其余所有表做出相同处理。
3. 数据库维护人员也有责任确保表之间的数据完整性同步。如果员工出现职务变动，必须确保数据完整且准确实现表间转移。

- **避免使用隐含或显式指明多个主题的名称。**这是最为常见的错误，解决它也相对容易。这类名称通常包含**和**与**或**以及斜杠 (/) 与并列符号 (&) 之类的符号；此类例子包括“部门或分支机构”以及“设施/建筑”。名称含义模糊表明，在分析和访谈的过程中，也许未能找准主题。可以翻看记录，如有必要，甚至开展进一步分析和访谈，从而解决这一问题。谨记必须始终确保每个表仅代表一个主题。

“杂项”这一名称可以归入这一范畴。上文中已经提到，这一名称所指难以言明；这种说法正确无误。不过，这一名称也暗含着多个主题；无法具体确认其主题，因为它似是而非，模糊不清。韦氏在线词典给出的定义如下：

杂项 adj. 1. 包含多种事物或成分；多元 2. 拥有各种特征

这一名称所造成的问题显而易见，所以绝不能将它作为表名称。

- **使用复数形式。**一个表代表一个主题，可指物品或事件。更进一步说，一个表可以表示某类物品或事件的一个集合。例如，一位销售代表想要维护自己所有顾客的数据，而不是单独的哪一个；一家汽车租赁企

业想记录所有车辆，而不只是那辆蓝色的宝马车。表名称使用复数形式看似不错，因为它表明了这是一个集合。当然，集合采用复数形式（是“Boats”，而不是“Boat”）。相反，识别字段的词语始终使用单数形式（“Home Phone”，而非“Home Phones”）。遵照这条规则，就能轻易区分所有文件材料中的表名称和字段名称。（重新制定表名称时，记住一些英语单词的复数形式并不以 *s* 或 *es* 结尾。例如，“equipment”的单复数形式完全相同。）

制定初始表列表中表名称应遵循以上指南。完成后的列表即为最终表列表，也是数据库设计过程发展至此的最终版本。注意，“最终”只指加入了整个分析过程中所发现的所有表。根据关系、数据完整性或其他发掘出来的信息，极有可能加入新的表。

❖注意：表名称使用复数形式的意见是非常有用的，特别是在数据库逻辑设计的时候。这样，就能轻易区分表名称和字段名称，使用投影屏幕展示或将它们写在会议室的白板上时尤其管用。

但要注意，当你（或负责实现该数据库的数据库开发人员）在 RDBMS 应用中实现该数据库时，表名可能改变。此后表名须遵照该 RDBMS 的一般约定。

指明表类型

本章前面已经提到，在最终表列表中应指明表类型。回想一下，表类型包括四种：数据、联系、子集和验证。

当你首先创建最终表列表时，列表上的每一项都是数据表，因为它们都代表重要主题，是数据库所提供信息的主要基础。列表中没有联系表和验证表，因为尚未建立关系，也未实施数据完整性。（在后面的部分，这些问题都会得到解决。）列表中也不会包含子集表，原因是在将字段分配到数据表中之后，才会建立子集表。

此时，指定最终表列表中每一个表为数据表。随着设计过程的展开，你

将指明其他类型的表。

编辑表描述

表描述是记录在最终表列表上的表的另一方面。表描述至关重要，因为它说明了表存在以及企业收集该表中数据的原因。实际上，描述必须**对表进行明确定义**，并且**阐明其对企业的重要性**。至于定义和重要性的先后顺序或是否只用一句话概括，都无关紧要。关键是，定义和重要性必须都包含在内。表描述也为验证表是否必要提供了一种手段。如果无法解释一个表对企业的重要性，就需要确认该表的时间和过程，并判断其存在的必要性。

既然本书为制订表名称提供了指南，在这里也为如何制订言简意赅、清晰明了表描述提出若干意见。

编辑表描述的指南

- **对表进行准确定义**。确保表描述中对表的定义简单明了，避免出现歧义和不确定之处。下面以一个“供应商”的表为例，该表取自一家面包店的数据库。

供应商——向我方供应材料和设备的公司

可以看到，以上定义并不准确。假如这家面包店也从当地农民处购进材料，这又该如何解释呢？毫无疑问，农民并不属于“公司”。这些供应商提供什么样的设备呢？炊具，手推车还是输送架呢？下面列出更为准确的定义：

供应商——向我方售卖材料和设备的公司和个人

以上定义可作为表描述中的**表定义**。

- **解释该表对企业的重要性**。出于特定的原因，企业为一个表收集数据，并负责维护，对其操作和检索。必须解释相应数据对企业重要的原因。记住，这个原因是组成表描述的一部分。可能做出如下陈述：

我们要用供应商表记录所有供应商的名称、地址、电话号码以及联系人姓名。

以上陈述有不当之处，它只强调了供应商表应保存的内容，而没有扩展到这些数据对企业重要的原因。下面的例子则更好地阐述了其重要的原因：

供应商信息之所以关键，是因为它保证了材料的持续供应，确保设备始终正常运转。

上述表述更为到位。通过指明供应商向面包店提供的服务，显示了这些数据的重要性。它也暗示没有供应商的服务，面包店将缺乏材料，其设备也难以正常运作。从而反映出该表重要的原因。

- **描述务求简明扼要。**避免出现对表名称的重复叙述或介绍，示例如下：

学生课表——学生课程安排表

避免过于简略或烦琐。确保大家都能识别表并理解其重要性的同时，还要避免信息过于复杂。下列示例就具有冗长、信息量过大的缺点：

学生课表——整个学年中，学生参加的所有课程（包括天数、时间以及授课人员）。本表中数据重要的原因是，学生从中得知课程名称及其地点、时间。此外，学生也能了解到课程的时长，以及授课教师的姓名。

以上示例可做如下改动，显得更为简明：

学生课表——所有课程系该学生本学年规定参加课程。本表旨在帮助学生更为有效地管理时间，让学校能够计算课程负荷和学生负荷。

第一句话对此表下了定义，第二句话说明了此表对该教学机构重要的原因。

- **避免提及具体操作信息，比如该表使用的方式和适用场合。**切勿叙述具体如何使用该表，以及如何访问该表。这类信息与数据库实现过程密切相关，而与本书中所介绍的数据库设计过程完全独立开来。下列示例展示的就是这种不当的描述：

学生课表——所有课程系该学生本学年规定参加课程。本表信息供教务长使用，经由登记程序中招生菜单访问。

- 不同表描述之间保持独立。每个表描述都应自成体系，独立于其他表描述；切勿将一表描述与另一表描述互相参照。不妨看看下列示例中的错误：

家属——配偶、子女以及该员工负责监护的未成年人。（详细信息参见员工表。）

改动如下：

家属——配偶、子女以及该员工负责监护的未成年人。本表为该员工减免税款及其所登记的福利计划之必要依据。

- 避免使用示例。示例是一种有效的交流工具，有助于传达特定含义或概念。巧妙地使用示例能达到事半功倍的效果。但是，示例需要依赖补充信息（并且在某些情况下，需要进一步的举例），才能传达预期的想法。不妨想想为了完整诠释表的主题，你所使用的一些示例。定义明确的描述简单明了，点到为止；因此，无须使用示例。

用户和管理人员访谈

制订表描述还需要开展用户和管理人员访谈，获得他们对表定义及其重要性的建议。（实际上，可以对用户和管理人员一起访谈。）主要目标是就表的总体描述达成一致。访谈完成后，保留记录并制订最终表描述，确保它遵循本章前面提到的指南。然后，再次同两方面人员交换意见，直至研究出大家一致认可的最终表列表。

考虑一下这个例子：假设你在为一家当地的软件培训机构开发数据库，助手约翰正和机构内部人员开展访谈。具体说来，和他对话的是行政部的马克、指导协调员弗里茨、销售副总裁萨拉以及机构负责人卡罗琳。下列对话为访谈节选部分。约翰当前讨论的是学生表。

与分析和设计过程的需求评审阶段所开展的访谈不同，无须对企业所有人员进行访谈。在后续设计过程中，只需与用户和管理人员的代表展开对话。

约翰：“好了，我们来讨论一下学生表吧。各位是如何定义学生的呢？”

弗里茨：“学生就是来参加我们的课程的私人。”

萨拉：“不全对。学生中也有企业派遣来参加我们课程的。例如，我们许多学生就来自当地银行和保险公司，这些机构为他们负担学费。”

马克：“没错，是这样。我想干脆就说学生是指前来参加我们课程的个体。”

（约翰记下了马克的总结。）

约翰：“好的，我懂了。大家都同意马克的意见吗？”

（所有人都点头赞成。）

“好极了。现在来说说，你们会如何解释学生信息重要的原因吧。”

卡罗琳：“没学生，我们哪来的业务呢？”

弗里茨：“如果我们对参加课程的学生进行记录，我们就可以向他们发送新课程方面的信息。”

萨拉：“记录这些信息能让我们及时更新账单和联系信息。对于将员工派遣到我们这儿上课的公司尤其如此。培训协调员可能出现职务变更，我们必须清楚接待的新面孔是谁。”

约翰：“说得好。还有别的吗？没有了？好的，大家都同意之前所说的吗？”

（所有人都再一次点头同意。因为没有人需要补充，约翰匆匆记下笔记，继续讨论下一个表。）

如你所见,开展这种访谈方式相当直接。注意,在确认大家没有什么需要补充之后,约翰征得了大家的一致同意。然后,他记录了要点,接着继续下一话题。

约翰完成访谈之后,他依据访谈记录为最终表列表中的每个表制订了表描述。为了使表描述符合要求,他必须解读和研究参与者反映的信息。基于检查,约翰写下如下描述:

学生——参加我方课程的个体。学生表中的数据进一步推销了本机构的课程,实现了与学生之间的适度沟通。

约翰接着为每个表编写描述。完成后,他再次与马克、弗里茨、萨拉和卡罗琳进行了对话,确保描述中肯易懂。

字段对应入表

第3章中已介绍了表是由字段组成的。在这一阶段,你要做的就是将初始字段列表中的字段分配到最终表列表的每个表。

将字段分配到表中是一个相对简单的过程:判断字段和表主题是否相符,再对号入座添加到相应表中。所有表逐一重复这个过程即可。如果发现某字段或某组字段可以表示多个表的特征,则分别做出相应的处理。等到精简表结构过程的时候,就能分辨表和字段是否对应正确。

❖注意:在接下来的示例中,特定程序将会用到纸笔。这样,在设计数据库时就能避免RDBMS程序的使用。需要强调的是,在数据库设计过程中,切勿使用计算机,除非是使用某种专门设计数据库的软件。这个建议旨在防止你陷入第14章“不良设计——禁忌”中提到的陷阱。

首先,取一张纸,横放在面前。在页首从左至右依次写下每个表的名称(取自最终表列表);表名称之间留出足够空间,以容纳长长的字段名称列表。然后,仿照这个过程完成最终表列表中剩余表的制作。再引用一下学校数据库的例子吧。图7.10所示为当前制作的表结构。

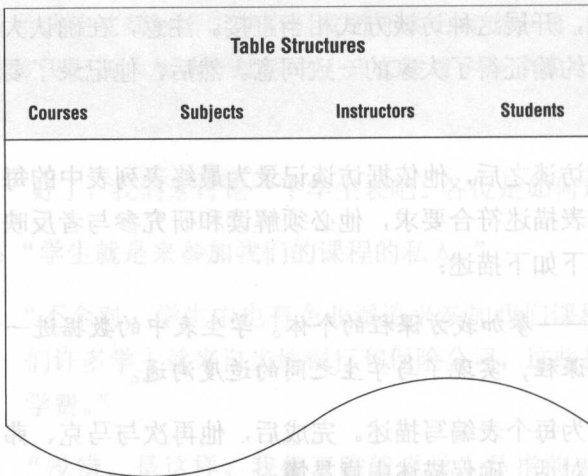


图 7.10 纸张上设立表结构

接下来，将初始字段列表中的字段分配到每个表中。判断哪些字段与该表主题相符，然后将对应字段添加到该表名称下。根据自己判断分配好字段后，继续对下一表重复这一过程，直至完成所有表。图 7.11 所示为表结构局部图。

Table Structures			
Classes	Subjects	Instructors	Students
Class Number	Subject Name	Instructor Name	Student Name
Class Name	Subject Description	Instructor SSN	Student SSN
Subject Name	Category	Instructor Address	Student Address
Instructor Name	Credits	Instructor Phone	Student Phone
Room Number		Date Hired	
		Pay Rate	

图 7.11 字段与表相对应

❖注意：在阅读下文之前，先回顾一下引言中提到的一条原则：

重点在于概念或技巧及其预期结果，而不是用来举证的例子。

之所以在此再次提到，是因为读者一定会疑惑为什么本书采用一种特殊的举例方式。也许，你会想到解决这个问题的更好途径，对此自有一套充分的解释。但是，切勿被示例所误导。本书采用特殊举例方式的唯一目的是阐明相应概念或技巧。因此，要着力研究示例中纠正问题的方式，确保在遭遇类似问题时能灵活运用这些技巧。

精简字段

既然已经将字段对号入座，现在就要改进字段名称并且解决可能存在的所有结构问题，从而精简字段。继而，通过确保字段与表正确对应和表结构完善，进一步精简表。

改进字段名称

上文提到，字段代表所属表主题的特征。如果某字段有与之相符的名称，就能轻易辨别该字段所表示的特征。字段名称模糊不清显然会造成麻烦，同时也表明并未完全理解该字段的意图。

本章前面介绍了为表命名的一系列指南。现在来介绍制订字段名称的若干指南。幸运的是，其中有多条与表命名的指南内容相近，所以你已经熟悉了大多数的概念。

制定字段名称的指南

- 字段名称应独特且富有内涵。一个字段名称只能在数据库中出现一次；唯一的例外就是建立起两表之间关系的字段。确保字段名称表述准确，词达其意。（第10章详细叙述了这个问题。）
- 字段名称应简明扼要，准确描述字段所代表的特征。“电话号码”这个例子就是典型的言辞含糊、定义不准。它所指的是什么电话号码呢？家庭电话？办公室电话？还是移动电话？应该具体一点。如果需

要记录这些号码,就可以分别创建“家庭电话”、“办公电话”和“移动电话”字段。

在第6章“分析现有数据库”中介绍了如何解决通用字段名称,比如“地址”、“城市”以及“州省”,方法就是使用相应表名称作为字段名称的前缀。这样就产生了诸如“*Employee Address* (员工地址)”、“*Customer Address* (顾客地址)”以及“*Supplier Address* (供应商地址)”之类的名称。如果存在这类字段名称,你可以使用前缀的缩写(为简略起见),即取英文名称前面三到四个字母作为前缀。这样,上述名称就变为了“*EmpAddress*”、“*CustAddress*”和“*SuppAddress*”。这一技巧不仅适用于本指南,也同样适用于上文中提到的指南。

❖注意:在表中是否使用前缀纯属个人习惯。如果表包含通用字段名称,一些数据库设计者只对通用名称使用前缀,而其他人则会对所有字段名称使用前缀。无论选择哪种方法,都应自始至终保持不变。

就个人而言,我更喜欢只对通用名称使用前缀,下文中也将始终采用这一方法。

- 字段名称应务求精简。避免拖沓冗长。例如,要记录员工加入企业的日期,“雇佣”太短(让人不知所云)，“聘用员工的日期”则过长!而“雇佣日期”较为合适,准确表达了字段所传达的特征。
- 切勿使用缩略语,慎用缩写词。缩略语可谓众口不一,往往造成误解。想想“CAD-SW”这个字段名称。如何判断这个字段所指为何呢?另一方面,应谨慎使用缩写词。除非缩写词有利于表达,否则应尽量避免。防止缩写词引发歧义或改变原意。
- 切勿使用混淆字段名称含义的词语。字段名称包含冗余成分或同义词,就会造成表意不明,引发误解。例如,“数字识别码号码”。“数字”和“号码”是多余成分,所以将之去除不会改变字段名称的含义。不妨假设你决定去除“数字”一词。这样,改动后的名称就可以分解为:“识别码”和“识别号码”。这两个名称区别不大,可以选择其中一个作为最终字段名称。不过其他情况下,就要使用最为恰当的名称。

- **避免使用隐含或显示多个特征的名称。**类似的名称易于发现，因为它们通常使用**和与或**。字段名称包含斜杠 (/) 或和号 (&) 也属此类。如果遇到的是“区域或地点”和“电话/传真”这样的名称，就先确认该名称所暗含的全部特征，再为每个特征分别创建一个新的字段。然后，对照这些指南检验字段名称，确保名称规范合理。
- **使用名称的单数形式。**字段采用复数形式的名称，比如“Skills(技术)”，意味着包含该记录的多个值，这并不合理。(本章后面有更多的介绍。) 字段名称之所以采用单数形式，是因为它表示所属表主题的**单个特征**。另一方面，表名称使用复数形式，原因在于它表示类似物体或事件的集合。根据这一命名规则，就能轻易区分表名称和字段名称了。

❖ **注意：**使用表名称作为字段名称的前缀，这种做法也会出现前面表名称提到的问题：一旦开始将数据库转化成 RDBMS 应用程序，这些特定的字段名称也可能发生改变。因为名称必须符合 RDBMS 中通用的命名规范。

掌握这些指南，逐一评审每个表，看看字段名称是否能有所改进之处。完成后，就可以识别并解决字段出现的任何问题。图 7.12 展示了对图 7.11 所示表结构的字段名称所做的改动。

在图 7.12 中，“Classes”简写为“Cls”，“Subjects”简写为“Subj”，“Instructors”简写为“Inst”，“Student”简写为“Std”，还有“Social Security Number”取代了“SSN”。记住，只要能表词达意，缩写词也是非常有用的。巧妙使用缩写词并不会影响字段名称的理解。

❖ **注意：**本章以及本书余下部分，语篇中提到的表名称所有字母都采用大写（例如 VENDORS），字段名称都采用首字母大写（例如 Vendor ID Number）。

Table Structures			
Classes	Subjects	Instructors	Students
ClsNumber	SubjName	InstName	StdName
ClsName	SubjDescription	InstSocial Security Number	StdSocial Security Number
SubjName	Category	InstAddress	StdAddress
InstName	Credits	InstPhone	StdPhone
Room Number		Date Hired	
		Pay Rate	

图 7.12 修正后的字段名称

使用理想字段解决异常现象

尽管仔细核查了初始字段列表中的字段，但是其中可能还是有一些会对表结构造成不利影响。字段定义不当就会产生重复数据和冗余数据，这些都是无法使用的。读者也许会觉得难以确认一个表中是否有问题字段，除非了解判定规则。确认潜在问题字段的最佳方法是判断字段是否具备理想字段的要素。这些要素都转化为了一系列的指南，可以指导创建规范的字段结构，也可以依据指南快速分辨出问题字段。

理想字段的要素

- **代表表主题的鲜明特征。**如你所知，表代表特定的主题，包括物体和事件。理想字段则表示该物体或事件的独特特征。
- **仅包含一个值。**可存储多个相同的值的字段称为多值字段。多值字段会造成数据冗余（这一点不言而喻），并且当你尝试编辑、删除或编排其中的数据时，会发现该多值字段无法使用。理想字段不存在这些问题，因为它只包含一个值。
- **无法分解为更小的元素。**一个值中可存储多个不同项的字段称为复合字段（也称合成字段）。与多值字段一样，这种字段会在编辑、删除或编排其中的数据时出现问题。理想字段之所以不会出现这些问题，是因为它表示所属表主题的一个独特特征。（本书随后即会更为详细

地介绍多值和复合字段。)

- **不含计算值或串联值。**同一表中字段的值应相互独立；特定字段的值应与其他字段的值无关。然而，计算字段的值却取决于其他字段的值，这便是问题所在。当与计算相关的字段值发生改变时，计算字段的值不能自动更新。这样，就需要用户或数据库应用程序对该计算字段进行更新。这就是单独处理计算字段的真正原因。
- **在整个数据库结构中独一无二。**在设计规范的数据库中，只有建立表之间关系的字段是重复字段。如果一个表中还存在其他重复字段，则该表极有可能累积了冗余数据，这些重复字段中数据也必定混乱。

❖ **注意：**记住，现阶段处理的是逻辑数据库结构。当你在 RDBMS 程序上实际操作时，可能出于特殊原因复制特定字段。不过，复制这些字段完全是有意意识的行为，你已经想好了应对措施。

- **主要特性始终保持不变。**若一个字段建立起两表之间的关系，则对于每个表而言，它都是表结构的组成部分。字段的主要特性始终保持不变。(第 9 章“字段说明”和第 10 章详细讨论了这个问题。)

尽管已经了解了理想字段的要素，但是仍会感觉到在许多情况下仅凭名称，难以辨别问题字段。图 7.13 所示表结构正说明了这一点。考虑一下，判断其中的字段是否符合理想字段的要素。

Table Structures	
Instructors	
InstName	
InstAddress	
InstPhone	
Categories Taught	
InstSocial Security Number	
Date Hired	
Pay Rate	

图 7.13 表中字段结构似有问题

列表中每个字段看似符合理想字段的要素。不过仔细检查列表，就会发现某些字段与第二和第三要素不符。InstName、InstAddress 和 Categories Taught 三个字段出现了异常，它们会引发问题。如果怀疑这一论断，可以把示例数据“加载”到表中进行检测。这样，很快就能发现异常问题（前提是存在异常问题），这也是确认字段是否符合理想字段所有要素的最佳方法。

不必真正创建一个表来执行这项验证。取一张纸，在案前平铺横放。在页首从左至右依次写下每个字段的名称；字段名称之间保留足够空间。然后，向表中输入记录，即在每个字段中填充一些样本数据；确保样本数据是即将输入数据库的数据。仅需少量记录就能顺利进行验证。图 7.14 所示即为公文用纸上的表。

Instructors					
InstName	InstAddress	InstPhone	CategoriesTaught	<< other fields >>	
Kira Bently	3131 Mockingbird Lane, Seattle, WA 98157	363-9948	DTP, SS, WP	
Timothy Ennis	7402 Kingman Drive, Redmond, WA 98115	527-4992	WP, DB, OS	
Shannon Black	4141 Lake City Way, Seattle, WA 98136	336-5992	DB, SS	
Estela Rosales	970 Phoenix Avenue, Bellevue, WA 98046	322-6992	DTP, WP, PG	

图 7.14 使用样本数据测试表

❖注意：第3章中已提到过，所展示的字段仅为与当前讨论最为相关的，<<other fields（其他字段）>>表示与此关系不大的字段。

现在，就能轻易分辨问题字段了。可以看出，InstName 和 InstAddress 是复合字段，Categories Taught 是多值字段。在精简表结构之前，必须解决这些字段。

消除复合字段

复合字段之所以无法使用，是因为它的值包含多个不同的元素。从复合字段中难以检索信息，也难以通过复合字段值对表中的记录分组或排序。图 7.14 中的 InstAddress 字段就存在这样的难题，检索西雅图的信息或根据邮政编码排序都会出现错误。

消除复合字段可以先确认字段值中的不同元素，再将每个元素单独作为一个字段。不妨先问自己一个简单的问题：“这个字段的值具体指的是什么？”一旦（尽己所能）想到了答案，分清了其中所含的元素，就将每个元素都转化为一个新的字段。

在图 7.14 中，InstName 的值表示两个元素：指导教师的姓和名。可以分别创建 InstFirst Name（指导教师的名）字段和 InstLast Name（指导教师的姓）字段。InstAddress 的值包含四个元素：街道地址、城市、州和指导教师的邮政编码。也要将这些元素转化为字段，分别为 InstStreet Address、InstCity、InstState 和 InstZipcode。图 7.15 所示为新修正后的 INSTRUCTORS 表。

Instructors

InstFirst Name	InstLast Name	InstStreet Address	InstCity	InstState	InstZipcode	InstPhone	CategoriesTaught	<< other fields >
Kira	Bently	3131 Mockingbird Lane	Seattle	WA	98157	363-9948	DTP, SS, WP
Timothy	Ennis	7402 Kingman Drive	Redmond	WA	98115	527-4992	WP, DB, OS
Shannon	Black	4141 Lake CityWay	Seattle	WA	98136	336-5992	DB, SS
Estela	Rosales	970 PhoenixAvenue	Bellevue	WA	98046	322-6992	DTP, WP, PG

图 7.15 修正后的 INSTRUCTORS 表

一些复合字段难以识别。看看图 7.16 中的 INSTRUMENTS 表。乍一看，表中似乎并不存在复合字段。然而，当更进一步检查表中的数据时，就会发现 Instrument ID 实际上是复合字段。这个字段的值表示两个元素：乐器所属的科目，包括 AMP(amplifier, 扩音器)、GUIT(guitar, 吉他)、MFX(multi-effects unit, 多效系统)、SFX(single-effect unit, 单效系统)，以及乐器的识别号码。显然，根据理想字段的第三要素应该将 Instrument ID 分解为两个字段。如果不予分解，一旦 MFX 类别更改为 MFU，更新字段值将何其困难。必须编写程序编码解析这个值，检测 MFX 的存在，然后在该解析值中存在 MFX 的情况下，使用 MFU 进行代替。虽然并非无计可施，但是无疑要多费心思，如果数据库设计规范，根本就不必面对这种问题。

GUIT = 科目 ("Guitar")
2201 = 识别号码

Instruments			
Instrument ID	Manufacturer	Instrument Description	<< other fields >>
GUIT2201	Fender	Stratocaster
MF33349	Zoom	Player 2100 Multieffects
AMP1001	Marshall	JCM 2000 Tube Super Lead
AMP5590	Crate	VC60 Pro Tube Amp
SFX2227	Dunlop	Cry Baby Wah-Wah
AMP2766	Fender	Twin Reverb Reissue

图 7.16 “隐式”复合字段示例

消除多值字段

前面已经提到，多值字段可能存储多个相同的值。幸运的是，多值字段仅从外表即可分辨。字段名称往往采用复数形式，它的值中也几乎总包含了多个逗号。使用逗号是为了将值包含的项分开。

与复合字段相比，消除多值字段要更难，需要花费一番工夫。多值字段与复合字段的基本问题相同，见图 7.17 中 Categories Taught 字段明显可知。例如，无法检索教授特定科目（比如 WP）的信息，也不能对数据进行合理排序，最重要的是，空间不足以容纳四个科目。如果有指导教师教授五个科目，那该怎么办呢？唯一的选择就是每次需要输入更多的值时，扩展字段。

逗号用于将多个项分开。

Instructors					
InstFirst Name	InstLast Name	InstStreet Address	InstCity	<< other fields >>	CategoriesTaught
Kira	Bently	3131 Mockingbird Lane	Seattle	DTP, SS, WP
Timothy	Ennis	7402 Kingman Drive	Redmond	WP, DB, OS
Shannon	Black	4141 Lake CityWay	Seattle	DB, SS
Estela	Rosales	970 PhoenixAvenue	Bellevue	DTP, WP, PG

图 7.17 识别多值字段

因此，该如何消除这个多值字段呢？第一想法可能是为每个值分别创建一个新的字段，从而多值字段“扁平化”为多个单值字段。图 7.18 所示即为这种做法的结果。

不幸的是，这样的处理无济于事。这种结构会引起三个问题。

1. 检索科目信息将变得异常费力。如果某位用户想找到所有教授 WP 科目的指导教师，就一定要确保在每个科目字段中分别搜索这个值，因为无法保证 WP 始终存储在相同的字段。该用户可能忽略一位合格的指导老师，从而导致检索失败。
2. RDBMS 程序无法对数据进行合理排序。
3. 这种结构具有先天的不稳定性。在当前的状态下，表未能限制指导教师所能教授的科目数量；当出现教授科目超过三门的指导教师，就必须创建额外的科目字段。增加科目字段恰恰会带来前两个问题。

Instructors

InstFirst Name	InstLast Name	InstStreet Address	InstCity	<< other fields >>	Category Taught 1	Category Taught 2	Category Taught 3
Kira	Bently	3131 Mockingbird Lane	Seattle	DTP	SS	WP
Timothy	Ennis	7402 Kingman Drive	Redmond	WP	DB	OS
Shannon	Black	4141 Lake CityWay	Seattle	DB	SS	
Estela	Rosales	970 PhoenixAvenue	Bellevue	DTP	WP	PG

图 7.18 Categories Taught 字段扁平化

意识到将 Categories Taught 字段扁平化并不能解决问题，接下来的想法是遵照理想字段的第二要素处理，使之只包含一个值。尽管这个灵感是在对的方向上踏出了一步，但是仍然无法完全解决这一问题，因为会引出另一个问题：数据冗余。图 7.19 展示了这一做法的结果。注意，表中每个记录的 Categories Taught 字段中只有一个值。

Instructors

InstFirst Name	InstLast Name	InstStreet Address	InstCity	InstState	InstZipcode	InstPhone	CategoriesTaught
Kira	Bently	3131 Mockingbird Lane	Seattle	WA	98157	363-9948	DTP
Kira	Bently	3131 Mockingbird Lane	Seattle	WA	98157	363-9948	SS
Kira	Bently	3131 Mockingbird Lane	Seattle	WA	98157	363-9948	WP
Timothy	Ennis	7402 Kingman Drive	Redmond	WA	98115	527-4992	WP
Timothy	Ennis	7402 Kingman Drive	Redmond	WA	98115	527-4992	DB
Timothy	Ennis	7402 Kingman Drive	Redmond	WA	98115	527-4992	OS
Shannon	Black	4141 Lake CityWay	Seattle	WA	98136	336-5992	DB
Shannon	Black	4141 Lake CityWay	Seattle	WA	98136	336-5992	SS
Estela	Rosales	970 PhoenixAvenue	Bellevue	WA	98046	322-6992	DTP
Estela	Rosales	970 PhoenixAvenue	Bellevue	WA	98046	322-6992	WP
Estela	Rosales	970 PhoenixAvenue	Bellevue	WA	98046	322-6992	PG

图 7.19 根据理想字段的第二要素修改 Categories Taught 字段

Categories Taught 的值造成冗余数据的原因在于，必须为指导教师教授的每门科目复制其记录。这种冗余显然是难以让人接受的，所以必须另辟蹊径。

为了避免以上状况的出现，可以采用以下步骤解决多值字段。

1. 将该字段从表中移除，以之作为基础创建一个新的表。如有必要，根据之前所学的字段命名指南为其重新命名。
2. 从原表中采用一个（或一组）字段建立起原表与新表的联系；尽量选取最能表示表主题的字段。所选取的字段同时出现在两表中。（第10章对关联表做了详细的介绍。）
3. 为新表制订合适的名称、类型和描述，并添加到最终表列表中。

下面介绍一下可以解决表中所有多值字段的一般方法。这样，也可以按照这些步骤处理 Categories Taught 字段。

1. 从 INSTRUCTORS 表中移除该字段，以之做为基础创建一个新的表。因为它现在变为了一个单值字段，所以重新命名 Categories Taught 字段。
2. 使用 InstFirst Name 和 InstLast Name 作为关联字段，建立起 INSTRUCTORS 表和新表的联系，并将这两个字段添加到新表的结

构中。

3. 为新表制订合适的名称，编写恰当的描述，并将该表添加到最终列表中。（标明表类型为“数据”。）下面列出了新表可以使用的名称和描述：

指导教师科目——系该指导教师胜任的软件程序科目。本表提供的信息旨在确保每门软件科目都配有足额指导教师。

图 7.20 所示为修正后的 INSTRUCTORS 表和新的 INSTRUCTOR CATEGORIES 表。

Instructors

InstFirst Name	InstLast Name	InstStreet Address	InstCity	InstState	InstZipcode	InstPhone
Kira	Bently	3131 Mockingbird Lane	Seattle	WA	98157	363-9948
Timothy	Ennis	7402 Kingman Drive	Redmond	WA	98115	527-4992
Shannon	Black	4141 Lake CityWay	Seattle	WA	98136	336-5992
Estela	Rosales	970 PhoenixAvenue	Bellevue	WA	98046	322-6992

Instructor Categories

InstFirst Name	InstLast Name	Category Taught
Kira	Bently	DTP
Kira	Bently	SS
Kira	Bently	WP
Timothy	Ennis	WP
Timothy	Ennis	DB
Timothy	Ennis	OS
Shannon	Black	DB
Shannon	Black	SS

图 7.20 消除 INSTRUCTORS 表中的多值字段后的结果

注意，新的 INSTRUCTOR CATEGORIES 表不再受多值字段的困扰，因为 Categories Taught 是一个单值字段。检索特定指导教师或科目的信息可以轻易实现，同时可以对记录进行合理排序。此外，InstFirst Name 和 InstLast Name 字段在新表中名称不变，符合理想字段的第五要素。

尽管新的表包含冗余数据，但这种冗余是可以接受的、最低限度的。实际上，关系数据库始终包含一定量的冗余数据。作为数据库设计师，目标就是确保冗余数据保持最低水平。

图 7.21 所示的 INSTRUCTORS 表包含三个多值字段。

Instructors

InstFirst Name	InstLast Name	Campus Phone	Categories Taught	Maximum Level Taught	Languages Spoken
Kira	Bently	363-9948	DTP, OS, SS, WP	Intermediate, Basic, Advanced, Basic	French, Spanish
Timothy	Ennis	527-4992	DB, OS, U T, WP	Intermediate, Basic, Basic, Advanced	German, Spanish
Shannon	Black	336-5992	DB, PG, SS	Advanced, Intermediate, Intermediate	French, German
Estela	Rosales	322-6992	DTP, PG, WP	Basic, Intermediate, Basic	French, Italian, Spanish

图 7.21 包含三个多值字段的 INSTRUCTORS 表

Categories Taught——表明该指导教师可教授的课程科目。

Maximum Level Taught——表明指导教师可教授该科目的最高等级。

Languages Spoken——表明指导教师掌握的外语。

现在，读者的任务就是使用刚刚学到的技巧解决这三个多值字段。结果发现了一个隐蔽的小问题：对于任意记录，Categories Taught 和 Maximun Level Taught 的值有一对一的关系。如果没有仔细检查这些字段中的样本数据，也许注意不到这种异常现象。不要担心；解决问题的步骤仍然不变，不过有一点小改动。

偶尔会遇到这种情况，即给定字段（无论单值或多值）取决于特定的多值字段。解决的方法很简单：将因变字段包含到为解决该多值字段所创建的新表中。图 7.22 所示即是按照这种方法解决上图中 Categories Taught 字段的结果，同时也展示了 Languages Spoken 的解决办法。

Instructors

InstFirst Name	InstLast Name	Campus Phone
Kira	Bently	363-9948
Timothy	Ennis	527-4992
Shannon	Black	336-5992
Estela	Rosales	322-6992

Instructor Categories

InstFirst Name	InstLast Name	Category Taught	Maximum Level
Kira	Bently	DTP	Intermediate
Kira	Bently	OS	Advanced
Kira	Bently	SS	Basic
Kira	Bently	WP	Advanced
Timothy	Ennis	DB	Intermediate
Timothy	Ennis	OS	Basic
Timothy	Ennis	UT	Basic
Timothy	Ennis	WP	Advanced

Instructor Languages

InstFirst Name	InstLast Name	Language Spoken
Kira	Bently	French
Kira	Bently	Spanish
Timothy	Ennis	German
Timothy	Ennis	Spanish
Shannon	Black	French
Shannon	Black	German
Estela	Rosales	French
Estela	Rosales	Italian
Estela	Rosales	Spanish

图 7.22 解决 INSTRUCTORS 表的多值字段

新表中的冗余保持最低水平，也是可接受的。第 10 章将介绍如何借助主键和外键建立表之间的联系，更进一步减少这种冗余。

精简表结构

既然已经精简了字段并确保每个字段合乎规范，现在可以开始精简表结构了。这一阶段的目标是确保字段合理分配到每个表和表的结构定义规范。这个过程也将揭晓表是否存在异常问题。

谈谈冗余数据和重复字段

本章中冗余数据一词使用十分频繁。在多数情况下，冗余数据都被贴上不可接受的标签，但是少数情况是允许的。为了更清楚该如何判断什么情况不允许有冗余数据，在此说明一下冗余数据的定义。

冗余数据是在某字段中出现多次的一个值，它的产生有两种情况：所属字段建立起两表之间的联系；某些字段或表出现异常。在第一种情况中，冗余字段是允许的；根据定义，一个字段建立起两表之间的联系，就会包含冗余字段。（详情参见第10章。）但是，第二种情况中的冗余数据是完全不可接受的，因为它会给数据一致性和数据完整性带来不利影响；因此，应该尽力使冗余数据保持在最低的水平。

重复字段就是在多个表中重复出现的字段，其成因很多。

- 用于将一组表联系起来。
- 表明特定类型的值出现多次。
- 对附加信息有需求倾向。

唯一允许出现重复字段的情况是在用来建立两表之间的关系时；在别无他法的情况下，才能采用重复字段将一表中的记录与另一表中的记录联系起来。除此之外，都可以排除重复字段。应该尽量避免使用重复字段，因为会产生无用的冗余数据。

在精简表结构的同时，评估是否应该保留表中的重复字段。如果存在的理由非常充分，就保留下来；否则，将之清除，下一节将介绍如何有效处理冗余数据和无用的重复字段。

参照理想表精简表结构

不管为精简表中的字段付出多少努力，都不能忽视表结构本身也许存在异常问题，会产生不必要的冗余数据而难以使用表中的数据。通过对比表是否具备理想表的要素，就可以判断表结构是否隐藏着问题。这些要素中蕴含了一套指南，可以指导创建规范的表结构，也可以轻松辨认出设计欠佳的表。

理想表的要素

- **表示单个主题**，可以是一个对象或事件。不错，这一点已多次提到。实际上，再多的强调也不为过。只要确保每个表只表示一个主题，就极大地降低了破坏数据完整性的潜在风险。这一要素可以验证数据库

设计中分析和访谈阶段的工作，以及最近所做的工作。

- **拥有一个主键。**这一点很重要，原因有二：主键是表中每个记录的唯一标识，而且它在建立表关系中发挥关键作用（别无他意）。另外，它具有独特的特征，有助于实施和实现各个层次的数据完整性。如果未能为每个表指派一个主键，就会最终导致数据完整性出现问题。更多主键的详情参见第 8 章“键”。
- **不含复合字段或多值字段。**理论上，在精简字段结构的时候，就应该解决掉这些问题。不过，最后再评审一次字段是可行的，确保完全消除了这些无用字段。
- **不含计算字段。**尽管可能认为当前数据库结构不含计算字段，但是在精简字段过程中可能忽略了其中的一两个。现在正是再次评审表结构的好时机，确保清除所有遗漏的计算字段。
- **不含无用的重复字段。**（注意，本条指南不适用于建立表之间联系的字段，比如图 7.22 所示示例中所用的字段。）设计欠佳的表的标志之一就是包含其他表中的重复字段。向表中添加重复字段的一个可能原因是，提供参考信息或表明特定类型值的多次出现。当使用该数据或设法从该表中检索信息时，这类冗余数据将会造成各种难题。
- **冗余数据量保持最低水平。**记住，关系数据库绝不可能完全摆脱冗余数据。不过，能够且应该尽量减少每个表的冗余数据量。

消除无用的重复字段

在对表结构做最后的修改前，必须先清除数据库中所有无用重复字段。然后，就能精简表，使之符合理想表的要求。

用于提供参考信息的重复字段也被称为**参考字段**。参考字段是多余的，不过也容易消除，可直接从表中清除。但是，许多人认为一个表必须包含报表中出现的所有字段，所以他们在表中加入了各种无用的重复字段。他们想当然地以为，这样表中就能够提供报表中的所有必要信息。不过，他们都大错特错，他们的行为既不理智又多余。表中包含参考字段显示其设计不佳，并且产生很多问题，其中许多会随着设计过程的展开愈发明显。用户或数据库应用程序必须确保多次出现的参考字段的值保持不变，而这个过程出错的

风险很高。图 7.23 所示示例为一个包含参考字段的表。

这些字段重复了 MANUFACTURERS 表中 MANPHONE 和 WEB SITE 字段。

Instruments							
Instrument ID	Instrument Description	Category	Price	Manufacturer	ManPhone	Web Site	
2201	Stratocaster	Guitar	\$ 799.99	Fender Musical Instruments	596-9690	www.fender.com	
3349	Player 2100 Multi-Effects	Multi-Effect Unit	\$ 174.99	Samson Technologies Corp.	364-2244	www.samsontech.com	
1001	JCM 2000 Tube Super Lead	Amplifier	\$ 549.99	Mesa/Boogie	778-6565	www.mesa-boogie.com	
5590	Crate VC60 Pro Tube Amp	Amplifier	\$ 399.99	St. Louis Music, Inc.	738-7563	www.crateamps.com	
2227	Cry Baby Wah-Wah	Single-Effect Unit	\$ 169.99	Dunlop Manufacturing, Inc.	745-2722	www.jimdunlop.com	
2766	Twin Reverb Reissue	Amplifier	\$ 1,224.99	Fender Musical Instruments	596-9690	www.fender.com	

Manufacturers							
Manufacturer	ManStreet Address	ManCity	ManState	ManZipcode	ManPhone	Web Site	
Dunlop Manufacturing, Inc.	PO Box 846	Benicia	CA	94510	745-2722	www.jimdunlop.com	
Fender Musical Instruments	8860 E. Chaparral Road	Scottsdale	AZ	85250	596-9690	www.fender.com	
Mesa/Boogie	1317 Ross Street	Petaluma	CA	94954	778-6565	www.mesa-boogie.com	
Samson Technologies Corp.	PO Box 9031	Syosset	NY	11791	364-2244	www.samsontech.com	
St. Louis Music, Inc.	1400 Ferguson Avenue	St. Louis	MO	63133	738-7563	www.crateamps.com	

图 7.23 一个包含参考字段的表

INSTRUMENTS 表中的 ManPhone 和 Web Site 字段是参考字段。根据定义，它们是无用的重复字段。无疑，表中不需要这两个字段，因为它们已经

是 MANUFACTURERS 表的一部分。因此，可以将它们从 INSTRUMENTS 表中清除，以解决无用重复的问题。（Manufacturer 不是一个参考字段，因为它连接着 INSTRUMENTS 表和 MANUFACTURERS 表。）第 12 章“视图”中将介绍如何将 INSTRUMENTS 表和 MANUFACTURERS 表融入一个视图（虚拟表），从而同时使用两表之中的字段。然后，就可以把这个视图作为基础，编写所需的报表。

用于表明同一类型值多次出现的重复字段也是不必要的。如图 7.24 中所示的 STUDENTS 表。

Students

StdFirst Name	StdLast Name	StdStreet Address	<< other fields >>	Instrument 1	Instrument 2	Instrument 3
Scott	Baker	2904 MadisonAve	Guitar	Tenor Sax	
Michael	Chow	7410 Taxco Drive	Tenor Sax	Clarinet	Electric Piano
Debbie	McGuire	332 158thAve SE	Drum Set	Bass Guitar	
Angie	Thomson	970 Pine Blvd	Guitar	Electric Piano	Snare Drum

这些重复字段表示的是相同类型的值出现三次。

图 7.24 包含无用重复字段的表

Instrument 1、Instrument 2 和 Instrument 3 是重复字段，表示相同类型的值多次出现。它们存在于表中的目的是，让音乐系能够记录学生通过考核的乐器。这些字段不仅提高了检索特定乐器信息的难度，而且限制了学生通过考核的乐器数量。如果一些学生想考核的乐器超过三种，又该怎么办呢？

这种字段结构看上去是不是有种似曾相识的感觉？没错！它和图 7.18 中的表有些相似。读者可能已经猜到，它不过是一个扁平化的多值字段。需要提醒的是，这个表的作者可能没想到这一点（许多创建这类字段的人也是如此），但是事实就是如此。

既然已经清楚如何消除多值字段，那么也应该知道如何处理这些重复字段。首先，将 Instrument 1、Instrument 2 和 Instrument 3 字段视为一个多值字

段，然后按照解决多值字段的方法进行处理。图 7.25 展示了这一过程。阴影中的 STUDENTS 表展示了如何将乐器字段当作一个多值字段。然后，运用之前学到的三步法解决多值字段，这样就有了修正后的 STUDENTS 表和新 STUDENT INSTRUMENTS 表。完成后，就能为学生输入任意数量的乐器。检索信息也将变得十分容易，比如考核通过吉他的学生姓名，当前某学生考核通过的乐器列表，以及通过电子钢琴的学生数量等。

Students							
StdFirst Name	StdLast Name	StdStreet Address	<< other fields >>		Instruments		
Scott	Baker	2904 Madison Ave		Guitar, Tenor Sax		
Michael	Chow	7410 Taxco Drive		Tenor Sax, Clarinet, Electric Piano		
Debbie	McGuire	332 158th Ave SE		Drum Set, Bass Guitar		
Angie	Thomson	970 Pine Blvd		Guitar, Electric Piano, Snare Drum		

Students				Student Instruments		
StdFirst Name	StdLast Name	StdStreet Address	<< other fields >>	StudFirst Name	StudLast Name	Instrument
Scott	Baker	2904 Madison Ave	Scott	Baker	Guitar
Michael	Chow	7410 Taxco Drive	Scott	Baker	Tenor Sax
Debbie	McGuire	332 158th Ave SE	Michael	Chow	Tenor Sax
Angie	Thomson	970 Pine Blvd	Michael	Chow	Clarinet
				Michael	Chow	Electric Piano
				Debbie	McGuire	Drum Set
				Debbie	McGuire	Bass Guitar

图 7.25 解决无用重复字段示意图

在某些情况中，表中能包含多组重复字段，表示相同类型的值出现多次。图 7.26 所示 STUDENTS 表与图 7.24 中的有轻微差别；这个版本中包含了两组重复字段。读者或许会问，“我明明看到三组重复字段，为什么作者说是两组呢？”与你的看法有所不同，实际上 Instrument 1/Checkout Date 1 不包含重复字段。正好相反，Instrument 1/Instrument 2/Instrument 3 包含一组重复字段，Checkout Date 1/Checkout Date 2/Checkout Date 3 包含另一组重复字段。

Students							
StdFirst Name	StdLast Name	<< other fields >>	Instrument 1	Checkout Date 1	Instrument 2	Checkout Date 2	Instrument 3 Checkout Date 3
Scott	Baker	Guitar	09/26/01	Tenor Sax	09/28/01	
Michael	Chow	Tenor Sax	09/26/01	Clarinet	10/03/01	Electric Piano 10/16/12
Debbie	McGuire	Drum Set	11/14/01	Bass Guitar	11/20/01	
Angie	Thomson	Guitar	11/14/01	Electric Piano	11/14/01	Snare Drum 12/05/12

图 7.26 一个包含多组重复字段的表

读者可能意识到，这两组重复字段实际是两个扁平化的多值字段，可以运用前面例子中使用的方法。必须注意另外一个问题，即乐曲和考核日期之间的一对一关系。不过，这不成问题，因为之前处理过这种情况。如果将多值字段 Instrument 和 Checkout Dates 可视化处理，就会发现整体的表格结构和图 7.21 的非常相似。（在后一视图中，Categories Taught 和 Maximum Level Taught 字段之间存在一对一关系。）

图 7.27 展示了如何修改这个表。和前面一样，阴影部分的 STUDENTS 表显示的是如何将乐器和考核日期当作单一的多值字段。然后运用前面学到的三步法消除这些多值字段，就有了修正后的 STUDENTS 表和新的 STUDENT INSTRUMENTS 表。

既然已经熟悉了理想表的要素，现在就来评审和精简表结构。如果对某个表持有怀疑，就将其结构略图画在纸上，加入样本数据。然后，就能够解决由数据揭示出来的异常问题。

正如数据表表示单一主题，子集表也表示一个数据表的一个从属主题。子集表包含了与其所表示的从属主题密切相关的所有字段，它也包含了一个或多个源自对应数据表的字段。这些字段用于连接对应数据表和子集表。需要注意的是，子集表不包含表示本身和数据库表中特征的字段：这些字段必须保留在数据表中。

既然确定 INVENTORY 表描述三个主题（包括两个从属主题），就必须清除示例所列的字段，使之成为更理想表的第 2 个要素。然后，使用消除冗余法建立两个新的子集表，每个从属主题分别为一个新的子集表。具体步骤如下。表 7.1 具器公成成附，跟建品用韩各室公成下含后中表，中最做一数据

Students				
StdFirst Name	StdLast Name	<< other fields >>	Instruments	Checkout Dates
Scott	Baker	Guitar, Tenor Sax	09/26/12, 09/28/12
Michael	Chow	Tenor Sax, Clarinet, Electric Piano	09/28/12, 10/03/12, 10/16/12
Debbie	McGuire	Drum Set, Bass Guitar	11/14/12, 11/20/12
Angie	Thomson	Guitar, Electric Piano, Snare Drum	11/14/12, 11/14/12, 12/05/12

Students

Student Instruments

StdFirst Name	StdLast Name	StdStreet Address	<< other fields >>	StudFirst Name	StudLast Name	Instrument	Checkout Date
Scott	Baker	2904 Madison Ave	Scott	Baker	Guitar	09/26/12
Michael	Chow	7410 Taxco Drive	Scott	Baker	Tenor Sax	09/28/12
Debbie	McGuire	332 158th Ave SE	Michael	Chow	Tenor Sax	09/28/12
Angie	Thomson	970 Pine Blvd	Michael	Chow	Clarinet	10/03/12
				Michael	Chow	Electric Piano	10/16/12
				Debbie	McGuire	Drum Set	11/14/12
				Debbie	McGuire	Bass Guitar	11/20/12

图 7.27 消除 STUDENTS 表中的多组重复字段

建立子集表

精简表结构的同时，会发现某个表中一些字段并非始终都包含值。这种情况不会影响从表中检索信息，但是它表明该表可能需要进一步改进。考虑一下图 7.28 中的 INVENTORY 表结构。

Table Structures	
Inventory	
Item Name	Model
Item Description	Warranty Expiration Date
Current Value	Publisher
Insured Value	Author
Date Entered	ISBN
Manufacturer	Category

图 7.28 办公清单表的结构

在这一场景中，表中包含了办公室各种用品的数据，例如办公器具、办

公设备（计算机、打印机等）以及书籍。不可避免，许多记录中多个字段的值是空白的。比如，一本书没有 Manufacturer（生产商）、Model（模型）和 Warranty Expiration Date（保修日期），打印机没有 Author（作者）、Publisher（出版者）、ISBN（国际标准书号）和 Category（科目）。从物理角度来说，这当然无关紧要（硬盘空间大小的限制不再像过去一样严重），不过它会让人感觉不对劲。用户（还有管理人员）看到表中出现大量空白，就会感到十分紧张。是不是遗漏了数据呢？难道有人忘记向这些字段输入数据了？或者有人误将数据删除了？还是计算机不慎将初始值破坏了？（没错，“计算机干的！”这个荒诞的谣传仍然存在。）更重要的问题是，如果秉承理想表的要素创建这个表，这种特殊的结构又是如何产生的呢？

幸运的是，这只是另外一种结构异常，设计各种表时偶尔会出现。读者的任务就是学会如何合理解决它。

第一步是确认 INVENTORY 表是否满足理想表的第一要求（例如，“它表示单一的主题”）。一个表的字段中包含许多个空白值，通常（但并非一定）表示多个主题。想一想示例所列的两组字段，马上就会意识到它们表示表主题的两个不同方面。第一组字段描述设备的库存，第二组字段描述书籍的库存；此外，两种库存具有共同的特征，比如 Item Name、Item Description 和 Current Value。本质上，“设备”和“数据”都是主题，它们的存在也都依赖于 INVENTORY 表，但彼此又有一定联系。因此，它们是从属主题，读者将为它们分别创建一个子集表。

正如数据表表示单一主题，子集表也表示特定数据表的一个从属主题。子集表包含了与其所表示的从属主题密切相关的字段，它也包含了一个或多个源自对应数据表的字段，这些字段用于连接对应数据表和子集表。需要注意的是，子集表不包含表示本身和数据表共同特征的字段；这些字段必须保留在数据表中。

既然确定 INVENTORY 表描述三个主题（包括两个从属主题），就必须清除示例所列的字段，使之满足理想表的第一要素。然后，使用清除的字段建立两个新的子集表，每个从属主题分别为一个新的子集表。具体步骤如下。

1. 使用 Manufacturer、Model 和 Warranty Expiration Date 字段创建一个新的子集表，称之为 EQUIPMENT。
2. 使用 Author、Publisher、ISBN 和 Category 字段创建一个新的子集表，称之为 BOOKS。
3. 在两个表中分别添加 Item Name 字段；这个字段用于连接每个子集表和数据表。
4. 为每个子集表分别编写恰当的描述，并将它们添加到最终表列表中。每个表的类型都标为“子集”。

图 7.29 所示为两个新子集表的结构。

Table Structures		
Inventory	Equipment	Books
Item Name	Item Name	Item Name
Item Description	Manufacturer	Publisher
Current Value	Model	Author
Insured Value	Warranty Expiration Date	ISBN
Date Entered		Category

图 7.29 新子集表的结构

花点时间再次评审表结构。也许会发现，已经在无意识的情况下创建了子集表。结构大体相同的表一般都是子集表；而它们之间通常只有少量互不相同的字段。例如，图 7.30 所示为两个表结构的局部图。每个表分别表示同一主题的不同方面。

Table Structures	Table Structures
Full-Time Employees	Part-Time Employees
FTEFirst Name	PTFirst Name
FTELast Name	PTLast Name
FTEDate Hired	PTDate Hired
Salary Amount	Hourly Rate
Position	Skill Level
FTEStreet Address	PTStreet Address
FTECity	PTCity
FTEState	PTState

图 7.30 子集表局部图

这两个表都表示员工，不过每个表分别表示特定类型的员工。但是，要注意两个表有不少相同的通用字段：名、姓、雇佣日期、街道地址、城市以及州。这些字段都是重复无意义的，所以要精简表结构，解决这一问题。

精简之前未发现的子集表

当发现这样的子集表，可以按照下列步骤进行精简。

1. 消除子集表所有相同的字段，并使用这些字段建立新的数据表。
2. 确认新数据表表示的主题，然后为其制定合适的名称。
3. 确保子集表表示该数据表的从属主题，如有必要，修改子集表名称。
4. 为数据表编写合适的描述，然后将它添加到最终表列表中。标明表类型为“数据”。

图 7.31 所示为按照上述步骤对 FULL-TIME EMPLOYEES 和 PART-TIME EMPLOYEES 表处理后的结果。

Employees	Full-Time Employees	Part-Time Employees
EmpFirst Name	EmpFirst Name	EmpFirst Name
EmpLast Name	EmpLast Name	EmpLast Name
Date Hired	Salary Amount	Hourly Rate
EmpStreet Address	Position	Skill Level
EmpCity		
EmpState		

图 7.31 精简子集表的结果

现在，所有表结构都接近良好。不过，当学习主键、外键、关系和业务规则后，还需要更进一步进行精简。

案例分析

现在，将为迈克自行车行精简初始表列表。读者已经知道，第一步是评审初始字段列表，确认列表中字段揭示的主题。图 7.32 所示为该列表的部分样本。

Birth Date	Office Phone
Employee Name	Product Name
Employee Address	Category
Employee City	Unit Price
Customer Name	Invoice Number
Customer Address	Invoice Date

图 7.32 迈克自行车行的初始字段列表

仔细评审整个初始字段列表之后，确定列表中字段暗示了这些主题：顾客、员工、发票、产品和供应商。然后，将这些字段汇编成第一版的初始表

列表。

将当前初始表列表与分析阶段创建的主题列表合并，创建第二版初始表列表。合并两个列表时要记住下列步骤。

1. 解决两个列表中重复项。记住，两个列表中都出现过的项可以表示不同主题。当你发现这些项时，运用相应的技巧进行处理。
2. 解决表示主题相同、名称不同的项。确保每个表表示一个独特的主题。
3. 将剩余项合为一个列表。合成的列表即为第二版的初始表列表。

经过这些步骤的处理之后，初始表列表应如图 7.33 所示。

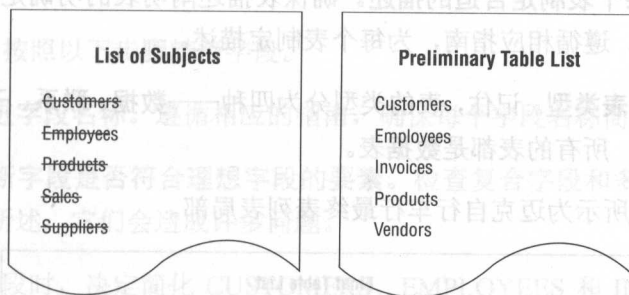


图 7.33 第二版初始表列表

之所以划掉主题列表中的“Customers”、“Employees”和“Products”，是因为它们与初始表列表中的对应项名称和所表示的主题都相同。虽然 SALES 表在初始表列表中无相同项，但它表示的主题与“**Invoices（发票）**”相同。不过，“**Invoices**”对于迈克和他的员工是最重要的，所以在初始表列表中采用它而不是“**Sales**”。“**Suppliers**”和“**Vendors**”的情况与之相似。迈克在初始表列表中选择了“**Vendors**”，所以划掉“**Suppliers**”。

❖注意：为表主题选取最恰当的名称应视具体情况而定。可遵循一条规则：使用大家最了解的名称。

现在，要制订最终版初始表列表。依据开始创建的任务目标，判断是否在前两个环节中有遗漏的主题。使用确定主题技巧找出任务目标中的所有主题。找到主题后，使用第二环节中的步骤，将这些主题与第二版初始表列表

中的主题进行对照检查。待完成检查并解决重复项，最终版初始表列表也就产生了。

结果显示，从迈克自行车行任务目标找到的所有主题都已出现在初始表列表中。这是一个好消息，因为对照检查将变得十分容易。待确信自己彻底完成任务，就有了最终版的初始表列表。

既然初始表列表已经完成，就要准备将它转化为最终表列表。随着这一过程的开始，要记住以下步骤。

1. **精简表名称。**遵循相应的指南，确保每个表名称简明扼要。
2. **为每个表制定合适的描述。**确保表描述阐明表的明确定义及其对企业的重要性。遵循相应指南，为每个表制定描述。
3. **标明表类型。**记住，表的类型分为四种——数据、联系、子集和验证。在这一阶段，所有的表都是数据表。

图 7.34 所示为迈克自行车行最终表列表局部。

Final Table List		
Name	Type	Description
Customers	Data	The people who purchase the products we have to offer. Keeping track of our customers allows us to promote our business and obtain valuable feedback in assessing the quality of our customer service.
Employees	Data	The people who work for our company in various capacities. This information is important for tax purposes, health benefits, and work-related issues.

图 7.34 迈克自行车行最终表列表局部

下一步是将初始字段列表中的字段分配到最终表列表的表中。确保选择的字段与对应表的主题最相符；每个字段应该定义或描述该主题的特定方面。图 7.35 所示为部分迈克自行车行表的结构。

Table Structures			
Customers	Employees	Invoices	Products
Customer First Name	Employee Name	Invoice Number	Product Name
Customer Last Name	Employee Address	Invoice Date	Product Description
Customer Phone	Employee Phone	Employee Name	Category
Customer Address	SSN	Customer Last Name	Wholesale Price
Status	Date Hired	Customer First Name	Retail Price
	Position	Customer Phone	Quantity

图 7.35 迈克自行车行表结构的部分列表

现在，按照以下步骤精简字段。

1. 改进字段名称。遵循相应的指南，确保每个字段名称简明扼要。
2. 判断字段是否符合理想字段的要素。检查复合字段和多值字段。如前所述，它们会造成许多问题。

评审字段时，决定简化 CUSTOMERS、EMPLOYEES 和 INVOICES 表中的一些字段名称，CUSTOMER 和 EMPLOYEE 简化为 Cust 和 Emp。PRODUCTS 表中的 Quantity 这个字段名称不能完全描述它所表示的特征，所以改为 Quantity On Hand。CUSTOMERS 和 EMPLOYEES 表中的电话字段也有同样的问题，所以 Custmer Phone 和 Employee Phone 分别改为 CustHome Phone 和 EmpHome Phone。此外，还将 SSN 改为 Social Security Number，使人一目了然。

通过对字段进一步的调查，发现几乎所有的字段都满足理想字段的要素。唯一的例外是 CUSTOMERS 和 EMPLOYEES 表中的地址字段，以及 EMPLOYEES 和 INVOICES 表中的 Employee Name 字段。经检查发现每个地址字段可以分解为四个独立的项：街道地址、城市、州和邮政编码，将这些项转化为字段，添加到 CUSTOMERS 和 EMPLOYEES 表中。同样，注意到 Employee Name 字段表示两项：姓和名，于是对 EMPLOYEES 和 INVOICES 表中该字段做适当的调整。

图 7.36 所示为调整后的结果。

Table Structures			
Customers	Employees	Invoices	Products
CustFirst Name	EmpFirst Name	Invoice Number	Product Name
CustLast Name	EmpLast Name	Invoice Date	Product Description
CustHome Phone	EmpHome Phone	EmpFirst Name	Category
CustStreet Address	Social Security Number	EmpLast Name	Wholesale Price
CustCity	EmpStreet Address	CustFirst Name	Retail Price
CustState	EmpCity	CustLast Name	Quantity On Hand
CustZipcode	EmpState	CustHome Phone	

图 7.36 字段改进后的表结构

最后的任务是精简表结构。确保字段与表正确对应，表的定义规范。具体步骤如下。

1. 消除无用重复字段。如果消除重复字段时创建了新的表，就要确保过程规范正确，并将它们添加到最终表列表中。
2. 判断每个表是否满足理想表的要素。确保解决在字段中或整个表结构内部发现的所有异常问题。
3. 酌情建立子集表。确保正确识别这些表，并添加到最终表列表中。

评审完表后，认定其他表都满足理想表的要素，只有 INVOICES 例外。这个表唯一的问题是，包含无用重复字段：CustHome Phone。不过，可以将这个字段直接清除，因为它提供的只是参考信息。

检查 PRODUCTS 表时，发现也许可以移除一些字段，用来创建一个子集表。所以再次评审该表。图 7.37 所示为当前检查的 PRODUCTS 表结构。（这是图 7.36 所示表结构的扩充版本。）

Table Structures	
Products	
Product Name	Service Name
Product Description	Service Description
Category	Service Category
Wholesale Price	Service Type
Retail Price	Materials Charge
Quantity On Hand	Service Charge

图 7.37 PRODUCTS 表结构（扩充版本）

事实证明设想是正确的。某些字段描述的是一种服务，可以将服务理解作为一种特殊的产品。服务与产品的相似之处在于都具有名称、描述和类别，不同之处则体现在类型、材料费用、服务费用和服务日期。出于这种考虑，创建了一个新子集表，称为 SERVICES（服务），对 PRODUCTS 表进行适当的修改，并使用 ProductName 字段连接两个表。然后，将 SERVICES 表相应的列表添加到最终表列表中。图 7.38 所示为修正后的 PRODUCTS 表和新的 SERVICES 子集表。

Table Structures	
Products	Services
Product Name	Product Name
Product Description	Service Type
Category	Materials Charge
Wholesale Price	Service Charge
Retail Price	
Quantity On Hand	

图 7.38 修正后的 PRODUCTS 表和新的 SERVICES 表

小结

本章以讨论初始表列表开篇。这个列表包含了新数据库的初始表结构。

文中介绍了如何使用分析阶段收集的初始字段列表、主题列表和目标任务开发这一列表。

接下来讨论了将初始表列表转化为**最终表列表**的过程，涉及数据库中每个表的名称、类型和描述。文中分别介绍了一套**创建表名称的指南**和一套**编辑表描述的指南**。然后，遵循相应的指南制订了简明扼要的表名称，也制订了规范的表描述，其中阐明了表定义及其对企业的重要性。读者也应意识到，获得用户和管理人员的帮助对开发定义明确的表描述也是至关重要的。表描述必须恰如其分，简单易懂。

然后，讨论的是为最终表列表中每个表分配合适的字段。文中介绍了如何从初始字段列表中选择最能代表表主题特征的字段，为给定表建立表结构。

接下来讨论了精简字段，文中介绍了一套创建字段名称的指南，确保字段名称言简意赅。另外，还介绍了理想字段的要素。通过判断某个字段是否满足这些要素，就能解决该字段中的异常问题。关于如何解决**复合字段**和**多值字段**，文中也做了详细说明。读者应知道，分解复合字段会产生新字段，分解多值字段则会产生新的表。

本章临近结尾讨论了精简表结构。文中介绍了理想表的要素。通过判断某个表是否满足理想表的要素，就能发现该表结构中隐藏的问题。随后讨论了**无用重复字段**，介绍了它出现在表中的一个原因：提供参考信息或表示相同类型的值出现多次。自然又引出了如何解决重复字段，消除它带来的问题。

最后，讨论的焦点集中于**子集表**。子集表表示特定数据表的一个**从属主题**，子集表和对应该数据表之间有一种独特的关系。文中介绍了何种情况下以及如何创建子集表。读者可能在不知情的情况下已经创建了子集表，那么就需要找出这些子集表。确认一个子集表后，精简该子集表，然后添加到最终表列表中。

思考题

1. 如何为新数据库识别并创建表？

2. 为什么定义表时要使用初始字段列表？
3. 主题列表与初始表列表中的项名称不同、表示主题相同，该如何处理？
4. 最终表列表提供哪些信息？
5. 说出创建表名称的三条指南。
6. 说出编写表描述的两条指南。
7. 如何将字段分配到最终表列表的表中？
8. 说出创建字段名称的三条指南。
9. 设计欠佳的字段会造成哪两个问题？
10. 使用什么解决字段异常？
11. 说出理想字段的三个要素。
12. 在何种情况下允许出现冗余数据？
13. 一般来说，遵循哪三个步骤解决多值字段？
14. 在何种情况下有必要在表中使用重复字段？
15. 怎样能精简表结构？
16. 说出理想表的三个要素。
17. 什么是子集表？

第 8 章 键

事实本身并无价值，有价值的是附着于事实的思想，或事实提供的证明。

——克洛德·贝尔纳

本章内容

键为何重要
为每个表建立键
表层次完整性
评审初始表结构
案例分析
小结
思考题

至此，读者已经标识了数据库将记录的所有主题，并定义了表示这些主题的表结构。此外，还精简了表结构，控制其构成和质量。在数据库设计过程的下一个阶段，将开始学习为每个表指定键。接下来，将了解不同类型的键及其在数据库结构中所发挥的独特作用。这一阶段将只指定一种键；指定其余的键则要等到以后建立表之间的关系（见第 10 章“表关系”）。

键为何重要

键对于表结构重要的原因如下。

- 确保准确识别表中每个记录。前面已经提到，一个表表示一个相似对象或事件的集合。（例如，CLASSES 表表示课程的集合，而不仅是一门课程。）这个集合是由表中一整套记录组成的，每条记录表示表主题的一个唯一实例。必须通过某些手段准确识别每个实例，键就是其中的一种。
- 有助于建立并实施各种完整性。键是表层次和关系层次完整性的主要部分。例如，键能确保一个表拥有唯一的记录和用于建立两表之间关系的字段始终包含匹配值。
- 用于建立表关系。第 10 章将介绍如何使用键建立两表之间的关系。

必须始终确保每个表都定义合适的键。这样，就能保证表结构完善，每个表中的冗余数据最少，以及表之间的关系稳固。

为每个表建立键

接下来的任务是为数据库中的每个表建立键。键的类型主要有四种：候选键、主键、外键和非键。键的类型决定了它在表中的功能。

候选键

为表创建的第一种键是候选键（candidate key），它是指可唯一识别表主题一个实例的一个或一组字段。每个表必须包含至少一个候选键。最终将检查表中的可用候选键，指定其中的一个为该表的正式主键。

将字段指定为候选键必须确保它满足候选键的所有要素。这些要素组成了一套指南，可以用于判断字段是否适合充当候选键。只要某个字段不符合任一要素，就不能将该字段指定为候选键。

候选键的要素

- **不得为复合字段。**上文中已对复合字段所带来的问题有过描述，因此复合字段不适于充当候选键。
- **必须包含唯一值。**这一要素能防止表中重复出现特定记录。重复记录犹如重复字段，必须尽最大可能避免。
- **不得包含 null 值。**文中早已提到，null 值表示无值。如果候选键的值为空，就绝不能识别记录。
- **其值不得违反机构安全或隐私规则。**字段拥有密码和社会保险号之类的值，不适于用作候选键。
- **其值无论部分或整体都不是可选值。**可选值意味着在某种情况下可能为 null。所以，它违反了前面的要素，不符合要求。（这一要素在选择多个字段作为候选键时尤其管用。）
- **包含定义唯一性所需的最少字段。**可以将一个字段组（看作一个单元）当作候选键，其中每个字段必须参与定义唯一值。不过，所需字段的数量应为最少，因为过于复杂的候选键最终将难以发挥作用、难以理解。
- **其值必须是识别表中每个记录的独特和唯一识别方式。**这一要素可避免重复记录，确保在数据库的其他表中能准确定位该表的任意记录。
- **其值必须是给定记录中每个字段值的唯一识别方式。**（主键一节中将对这一要素进行详细介绍。）
- **除非特殊情况，否则不得修改其值。**若无十分充足的理由，切勿修改候选键的值。随意修改某个字段的值，则该字段可能无法满足前述的要素。

为表建立候选键是轻而易举的：寻找满足所有候选键要素的单个或一组字段。对于给定的表，也许找出的候选字段不止一个。向表中加载样本数据，有助于准确识别潜在候选键。（在上一章中使用过这一技巧。）

看看是否能够为图 8.1 中的表找出几个潜在的候选键。

Employees

Employee ID	Social Security Number	EmpFirst Name	EmpLast Name	EmpStreet Address	EmpCity	EmpState	EmpZipcode	EmpHome Phon
1000	987-65-9938	Kira	Bently	1204 Bryant Road	Seattle	WA	98157	363-9948
1001	987-65-6531	Katherine	Erllich	101 C Street, Apt. 32	Bellevue	WA	98046	322-6992
1002	987-65-0039	Timothy	Ennis	7402 Kingman Drive	Redmond	WA	98115	527-4992
1003	987-65-1299	Shannon	Black	4141 Lake CityWay	Seattle	WA	98136	336-5992
1004	987-65-6529	Susan	Black	2100 MineolaAvenue	Seattle	WA	98115	572-9948
1005	987-65-5583	Estela	Rosales	101 C Street, Apt. 32	Bellevue	WA	98046	322-6992
1006	987-65-1734	Timothy	Sherman	66 NE 120th	Bothell	WA	98216	522-3232

图 8.1 表中是否有候选键?

读者可能认为 Employee ID、Social Security Number、EmpLast Name、EmpFirst Name 与 EmpLast Name、EmpZipcode 和 EmpHome Phone 是潜在的候选键。但是, 需要更细致的检查, 才能分辨哪些是真正的候选键。记住, 凡违反候选键任一要素的字段都要排除。

根据细致的检查, 你得出以下结论。

- Employee ID 合格。这个字段满足候选键的所有要素。
- Social Security Number 不合格, 因为它包含 null 值, 极有可能危害到机构的隐私规则。与样本数据所示结果相反, 这个字段可能包含一个 null 值。例如, 在美国许多参加工作的人都没有社会保险号, 因为他们其他国家公民。

❖注意: 尽管在许多类型的数据中广为使用, 本书还是强烈建议不将 Social Security Number 作为任何数据库结构中的候选键 (或主键)。在许多情况下, 它不满足候选键的要素。

Social Security Online 网站上的 Philadelphia Region 部分提供了一些关于 Social Security 号码和身份盗用的有趣事实, 这是应避免使用 SSN 作为主键的另一原因。地址为 www.ssa.gov/phila/ProtectingSSNs.htm。

- EmpLast Name 不合格, 因为它可包含重复值。前面已经提到, 候选键的值必须为唯一值。在本例中, 特定姓氏出现了多次。
- EmpFirst Name 与 EmpLast Name 合格。两个字段的组合值为特定

记录提供了唯一的识别方式。尽管有些姓和名出现多次，但是其组合却始终是一致的。（有些人可能会说，“这种说法也不正确。”没错，不过不必担心；这个问题马上就会得到解决。）

- **EmpZipcode** 不合格，因为它可包含重复值。许多人的住所共用一个邮政编码，因此 EmpZipcode 字段的值不是唯一值。
- **EmpHome Phone** 不合格，因为它可包含重复值，并且它的值可改变。这一字段包含重复值的原因可为下列任意一个。
 1. 多个家庭成员任职于该机构。

2. 多人同住共享一部电话。

可以认定 EMPLOYEES 表中有两个候选键：Employee ID 和 EmpFirst Name 与 EmpLast Name 的组合。

在指定为候选键的字段名称旁写下“CK”，标明表结构中的候选键。包含多个字段的候选键称为复合候选键（composite candidate key），在组成这种键的字段名称旁写下“CCK”。如果存在多个复合候选键，则在“CCK”标志后加注编号予以区分。比如有两个复合候选键，就可以将其中一个标为“CCK1”，另一个标为“CCK2”。

将这一技巧运用到图 8.1 的 EMPLOYEES 表中。图 8.2 所示即为加上标志后的结构。

Table Structures	
Employees	
Employee ID	CK
Social Security Number	
EmpFirst Name	CCK
EmpLast Name	CCK
EmpStreet Address	
EmpCity	
EmpState	
EmpZipcode	
EmpHome Phone	

图 8.2 EMPLOYEES 表结构中标记候选键

现在，找出图 8.3 所示 PARTS（零件）表中的候选键。

Part Name	Model Number	Manufacturer Name	Retail Price
Shimka XT Cranks	XT-113	Shimka Incorporated	199.95
Faust Brake Levers	BL / 45	Faust USA	53.79
MiniMite Pump		MiniMite	35.00
Hobo Fanny Pack		Hobo Bike Company	59.00
Diablo Bike Pedals	Mtn-A26	Diablo Sports	129.50
Shimka Truing Stand	SP-100		37.95
Faust Brake Levers	BL / 60	Faust USA	79.95

图 8.3 你能找出 PARTS 表中的候选键吗？

乍看之下，可能认为 Part Name（零件名称）、Model Number（型号）、Part Name 和 Model Number 的组合以及 Manufacturer Name（制造商名称）和 Part Name 的组合是潜在的候选键。不过，经过研究之后，你得出以下结论。

- Part Name 不合格，因为它可包含重复值。当给定零件具有多种型号，该零件名称就会重复。例如，Faust Brake Levers（福斯特制动杆）就是这种情况。
- Model Number 不合格，因为它可包含 null 值。候选键的值必须始终存在。从图中可以看出，部分零件没有型号。
- Part Name 和 Model Number 不合格，因为其中某一字段可包含 null 值。Model Number 可包含 null 值这一简单的事实，就使得这个字段组合不符合候选键的条件。
- Manufacturer Name 和 Part Name 不合格，因为这些字段的值是可选的。记住，候选键的值无论部分或整体都不是可选的。在这一情况中，零件名称中包含制造商名称，由此可推断输入制造商名称是可选的；因此，不能将这个字段组合制定为候选键。

显然，PARTS 表中不存在候选键。这个问题不由令人担忧，因为每个表必须拥有至少一个候选键。幸运的是，有一种解决方法。

人造候选键

当你判断一个表不含候选键时，可以创建和使用人造（artificial）或代理（surrogate）候选键。（人造是指该候选键不在表中“自然”出现；必须另外创建。）创建人造候选键的步骤是：创建一个满足所有候选键要素的新字段，然后将它添加到表中；这个字段就成为了正式候选键。

这样，就可以解决 PARTS 表中的问题了。创建一个人造候选键，称之为 Part Number，再将它指定给该表。（新的字段满足候选键要素，因为它是根据这些要素创建的。）图 8.4 所示为修正后的 PARTS 表结构。

Parts

Part Number	Part Name	Model Number	Manufacturer Name	Retail Price
41000	Shimka XT Cranks	XT-113	Shimka Incorporated	199.95
41001	Faust Brake Levers	BL / 45	Faust USA	53.79
41002	MiniMite Pump		MiniMite	35.00
41003	Hobo Fanny Pack		Hobo Bike Company	59.00
41004	Diablo Bike Pedals	Mtn-A26	Diablo Sports	129.50
41005	Shimka Truing Stand	SP-100		37.95
41006	Faust Brake Levers	BL / 60	Faust USA	79.95

图 8.4 拥有人造候选键 Part Number 的 PARTS 表

为一个表创建完人造候选键，在该表结构中该字段名称旁标注“CK”，做法和前面的 EMPLOYEES 表一样。

如果人造候选键比既有的所有候选键更为强大、更为合适，可能也会选择创建一个人造候选键。假设正在开发一个 EMPLOYEES 表，认定唯一可用的候选键是 EmpFirst Name 和 EmpLast Name 字段的组合。虽然这是一个有效的候选键，但是使用单字段候选键更为便捷，识别表中的主题更容易。恰好，机构内部的所有人都习惯使用独特的识别号码识别一名员工，而不是使用姓名。有鉴于此，可以选择创建一个新字段，称为 Employee ID，使用它作为人造候选键。这是一种完全可接受的做法——如果认为合适，毫无保留、不加犹豫地去做。

❖ 注意：本书通常创建一个 ID 字段（比如，Employee ID、Vendor ID、

Department ID、Category ID 等), 作为人造候选键。它有三个优点: 始终满足候选的条件, (最终) 能成为优秀的主键, 以及使得建立表关系更为容易 (详情见第 10 章)。

评审已选的候选键, 务必确保完全满足候选键的要素。如发现其中有不合格的候选键, 无须意外, 此类错误判断偶有发生。直接将该字段名旁的“CK”标志清除。只要表中候选键不止一个, 选择候选键就不成问题。不过, 如果发现表中确认的唯一一个候选键不合格, 就必须创建一个人造候选键。定义新的候选键后, 记得在表结构中其名称旁标注“CK”。

主键

至此, 已经为每个表建立了合适的候选键。下一任务就是为每个表建立一个主键, 它是所有键中最为重要的键。

- **主键字段**是整个数据库结构中识别其所属表的唯一方式, 也有助于建立与其他表之间的关系。(详情见第 10 章。)
- **主键值**是可识别表中特定记录的唯一方式, 也是整个数据库中表示该记录的专有方式。它能有效防止重复记录。

主键必须满足与候选键完全相同的所有要素。这一要求容易达成, 因为可以从表的有效候选键中选择主键。选择主键的过程与美国总统竞选有几分相似。每过四年, 总会有几个人参与竞选美国总统。这些人被称为“候选人”, 他们满足担任总统的所有条件。随着全国大选的举行, 总统候选人中就会有一人被选举为国家总统。同样, 确认好表中所有符合条件的候选键, 开展竞选, 然后选举其中一个作为正式的主键。确认工作已经完成, 现在就是竞选的时刻了!

如无特殊需要, 可根据下列指南选择主键。

1. 当单字段候选键和复合候选键同时存在时, 选择单字段候选键。包含最少字段的候选键始终都是最佳选择。
2. 选择名称中包含部分表名称的候选键。例如, 一个候选键的名称为

Sales Invoice Number, 对于 SALES INVOICES 表而言它就是一个出色的选项。

检查候选键, 从其中选择一个作为表的主键。选择标准因人而异。可以选择自己认为表达表主题最准确或者对机构全体人员最具意义的候选键作为主键。例如, 再考虑一下图 8.5 所示的 EMPLOYEES 表。

Table Structures	
Employees	
Employee ID	CK
Social Security Number	
EmpFirst Name	CCK
EmpLast Name	CCK
EmpStreet Address	
EmpCity	
EmpState	
EmpZipcode	
EmpHome Phone	

图 8.5 哪个候选键应成为 EMPLOYEES 表的主键?

从表中找出的所有候选键都不能作为主键。如果机构内部所有人都习惯在一些场合使用 Employee ID (工作证号码) 作为识别员工的方式, 比如纳税申报表和员工福利计划, 或许就会决定选择它。最终选择的候选键将成为该表的主键, 也要符合主键的要素。这些要素与候选键的要素完全相同, 应该严格予以遵行。为了表述明确, 在此列出主键的要素。

主键的要素

- 不得为复合字段。
- 必须包含唯一值。
- 不得包含 null 值。
- 其值不得违反机构安全和隐私规则。
- 其值无论部分或整体都不是可有可无的。
- 包含定义唯一性所需的最少字段。
- 其值必须是识别表中每个记录的独特和唯一识别方式。

- 其值必须是给定记录中每个字段值的唯一识别方式。
- 除非特殊情况，否则不得修改其值。

最终选定主键之前，务必确保主键满足以下要素：

其值必须是给定记录中每个字段值的唯一识别方式。

在整个数据库中，给定记录的每个字段值应为唯一值（除非它参与了建立两表之间的关系）且只有一种识别方式，即该记录的特定主键值。

遵循以下步骤可以判断主键是否完全满足这一要素。

1. 向该表中加载样本数据。
2. 选择一个记录用于测试，记录当前主键值。
3. 检查主键后第一个字段的值，考虑下列问题：

这个主键值是识别当前字段值的唯一方式吗？

- a. 是，继续下一字段，重复这一问题。
- b. 不是，将这个字段从表中清除，继续下一字段，重复这一问题。

4. 继续这一过程，直至检查完记录中的每个字段值。

如果主键不能唯一地识别一个字段值，则表明表结构中不需要该字段；因此，应该将该字段移除，再确认该表是否满足理想表的要素。然后，可以将刚刚移除的字段添加到另一个合适的表中，或者因为完全无用，直接丢弃。

图 8.6 中示例展示的是如何使用这项技巧处理表结构。注意，Invoice Number（发票号码）是表的主键。

Sales Invoices

Invoice Number	Invoice Date	CustFirst Name	CustLast Name	EmpFirst Name	EmpLast Name	EmpHome Phone
13000	06/15/02	Frank	DeSoto	Estela	Rosales	363-9948
13001	06/15/02	Gregory	Mattson	Katherine	Erich	322-6992
13002	06/15/02	Carmen	Aguilar	Kira	Bently	527-4992
13003	06/16/02	David	Cunningham	Kira	Bently	336-5992
13004	06/16/02	Carmen	Aguilar	Shannon	Black	572-9948
13005	06/17/02	Frank	DeSoto	Estela	Rosales	322-6992

图 8.6 表中主键是每个字段值的唯一识别方式吗?

首先,向表中加载样本数据。然后,选择一个记录用于测试(本书使用第三个记录作为示例),并记录主键的值(13002)。现在,就该记录中每个字段值考虑一下下列问题。

该主键值是当前××字段值的唯一识别方式吗?

Invoice Dare (发票日期) 是,发票号码始终可以识别该发票创建的具体日期。

CustFirst Name (顾客的名) 是,发票号码始终可以识别参与该交易的顾客的名。

CustLast Name (顾客的姓) 是,发票号码始终可以识别参与该交易的顾客的姓。

EmpFirst Name (员工的名) 是,发票号码始终可以识别负责该交易的员工的名。

EmpLast Name (员工的姓) 是,发票号码始终可以识别负责该交易的员工的姓。

EmpHome Phone (员工的家庭电话) 不是,发票号码不能通过员工姓名直接识别其家庭电话。实际上,同时需要 EmpFirst Name 和 EmpLast Name 的当前值才能识别 EmpHome Phone 的值,改变员工的姓名,员工电话也必须随之改

变。移除 EmpHome Phone 有两个原因：主键不是其当前值的唯一识别方式，而且（如你所料）它是无用字段。其实，你可以将之完全剔除，因为它早已是 EMPLOYEES 表的一部分。

移除不必要的字段后，检查修正后的表结构，确保它满足理想表的要素。

现在，该主键成为了识别表中剩余字段的值的唯一方式。这意味着，该主键达到完善，可以将它指定为正式的主键了。将表中该字段名称旁的“CK”替换成“PK”。（一个主键若包含多个字段，则称为复合主键。如果是复合主键，则替换成“CPK”。）图 8.7 所示为修正后 SALES INVOICES 表的结构，其中 Invoice Number 为其主键。

Table Structures	
Sales Invoices	
Invoice Number	PK
Invoice Date	
CustFirst Name	
CustLast Name	
EmpFirst Name	
EmpLast Name	
Ship Date	
Shipper Name	

图 8.7 修正后的 SALES INVOICES 表，其主键为 Invoice Number

为数据库中的每个表创建一个主键时，记住以下两条规则：

创建主键的规则

1. 每个表有且仅有一个主键。因为主键必须满足规定的所有要素，每个表只需有一个主键。
2. 数据库中每个主键必须为唯一的，即任意两个表不得有相同的主键，

除非其中一个为子集表。本节开首即介绍了整个数据库结构中主键为识别一个表的唯一方式；因此，每个表必须有其独特的主键，避免任何与表的识别性相关的混乱和模糊。这一规则之所以排除子集表在外，是因为它表示特定数据表主题的特殊版本，子集表与对应数据表必须共享同一主键。

在后续过程中，你将学习如何使用主键帮助建立两表之间的关系。

替换键

既然已经选择了特定表候选键作为主键，就将剩下的键指定为替换键 (alternate key)。在 RDBMS 程序中，这些键十分有用，因为它们为唯一识别表中的特定记录提供替换方式。如果选择将一个候选键作为替换键，就在其名称旁标注“AK”或“CAK (复合替换键)”；否则，无须任何处理，任其作为常规的字段。在后续的数据库设计过程中，不会涉及替换键，但是等到在 RDBMS 程序中实施数据库时将再次用到。(在 RDBMS 程序中实施和使用替换键不在本书涉及范围内，此处本书的唯一目的是指定适当的替换键。这与本书的中心是一致的，也就是数据库的逻辑设计。)

图 8.8 所示为 EMPLOYEES 表的最终结构，其中指定了合适的主键和替换键。

Table Structures	
Employees	
Employee ID	PK
Social Security Number	
EmpFirst Name	CAK
EmpLast Name	CAK
EmpStreet Address	
EmpHome Phone	
EmpCity	
EmpState	
EmpZipcode	
EmpHome Phone	

图 8.8 EMPLOYEES 表的最终结构

非键

非键 (non-key) 是不属于候选键、主键、替换键或外键的字段。它的唯一目的就是表示所属表主题的一个特征, 它的值由主键决定。非键无任何特殊标志, 所以不需在表结构中标记。

表层次完整性

这种完整性是整体数据完整性的主要部分, 它确保了以下优点。

- 表中无重复记录。
- 主键为表中每个记录的唯一识别方式。
- 每个主键值是唯一的。
- 主键值不为 null。

当定义每个表的主键时, 务必确保每个主键完全满足主键的要素, 保证其有效实施。这样, 就可以开始建立表层次完整性了。在下一章中, 建立表中每个字段说明的同时, 将进一步提升表的完整性。

评审初始表结构

既然表的基本定义已经完成, 现在就需要开展用户和管理人员访谈, 评审至今所做的工作。这一组访谈应该直切主题, 相对比较容易。

访谈时应完成以下任务。

- 确保合适主题在数据库中被表达。尽管在这一阶段遗漏重要主题是基本不可能的, 但是难免不虞之变。如果真有此事, 不妨先确认主题, 再运用相应技巧将之转化为表, 最后将该表开发成与数据库中其他表同步的水平。
- 确保表名称和表描述规范且准确易懂。如果表名称或描述让机构内部人员难以理解或者产生误解, 就要设法使之得到澄清。访谈过程中, 一些表名称和描述往往能得到改进。

- **确保字段名称规范且准确易懂。**选择字段名称通常能引发大量的讨论，特别是存在既有数据库的情况下。读者通常会发现人们习惯使用特定名称指代某一字段，因为“我的屏幕上就是这样叫的。”当出于充分的理由，改变一个字段名称时，必须向这些人委婉解释，为该字段重命名目的是满足新数据库的标准。也可以告诉他们，等到数据库在 RDBMS 程序上实施的时候，这个字段可能会有大家更为熟悉的名称。事实确实如此，许多 RDBMS 程序都允许使用一个名称作为字段的物理定义，另一个名称则用于显示。不过，这一特点并不影响遵循第7章“建立表结构”中的创建字段名称的指南。
- **核实已制定到表的所有字段。**确保所有与表主题相符的必要特征到位，这是最好的时机。通常会发现前面阶段不慎疏忽了一两个特征。如果确实如此，则一一确认这些特征，并使用相应的技巧将之转化为字段，再遵循所有必要的步骤把字段添加到表中。

访谈完成后，将进入下一阶段为数据库中每个字段建立字段说明。

案例分析

现在，该为迈克自行车行的数据库中每个表建立键了。如你所知，第一项任务就是为每个表建立候选键。不妨假设从图 8.9 所示 CUSTOMERS 表入手。

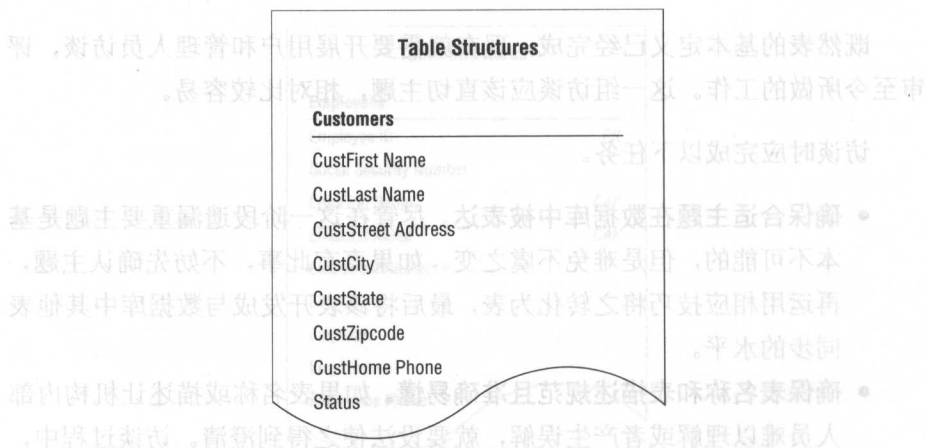


图 8.9 迈克自行车行数据库中的 CUSTOMERS 表结构

评审每个字段的同时，设法判断它们是否符合候选键的要素。虽然可以确定 Status、CustHome Phone 以及 CustFirst Name 和 CustLast Name 的组合是潜在的候选键，但不能确定它们是否都完全满足所有要素。因此，可以向表中加载样本数据，测试这些键，如图 8.10 所示。

Customers

CustFirst Name	CustLast Name	CustStreet Address	CustCity	CustState	CustZipcode	CustHome Phone	Status
Bridget	Berlin	2121 NE 35th	Bellevue	WA	98004	422-4982	Valued
Phillip	Bradley	101 9th Avenue	Kent	WA	98126	322-1178	
Kel	Brigan	7525 Taxco Lane	Redmond	WA	98225	363-9360	Valued
Barbara	Carmichael	7525 Taxco Lane	Redmond	WA	98225	363-9360	Preferred
Daniel	Chavez	750 Pike Street	Bothell	WA	98001	441-3987	Valued
Daniel	Chavez	301 N Main	Seattle	WA	98115	365-7199	
Sandi	Cooper	115 Pine Place	Seattle	WA	98026	332-0499	Preferred

图 8.10 用于测试候选键的 CUSTOMERS 表

记住，一个字段必须满足所有候选键要素，才能认定为候选键。如果不符合这一要求，应立即排除。

检查该表之后，得出以下结论。

- Status 不合格，因为它可能包含重复值。随着业务的扩展，迈克将拥有更多“重要 (valued)”顾客。
- CustHome Phone 不合格，因为它可能包含重复值。样本数据显示，两个顾客可能同住在一起，拥有相同的电话号码。
- CustFirst Name 和 CustLast Name 的组合不合格，因为它们可能包含重复值。样本数据显示，顾客中出现了同名同姓的现象。

这些发现让你确信需要为这个表创建一个人造候选键。于是，创建一个 Customer ID 字段。待确认它满足候选键的所有要求后，将这个新的字段添加到表结构中，并标注候选键标志。

图 8.8 所示为修正后的 CUSTOMERS 表的结构。

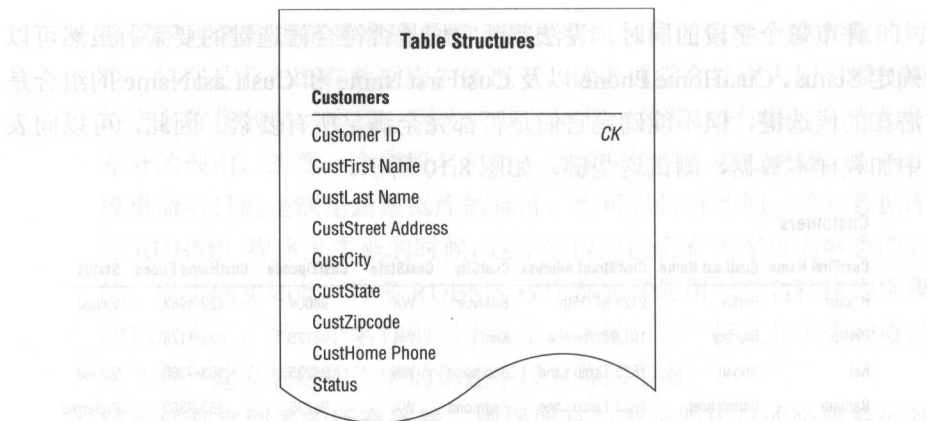


图 8.11 创建人造候选键后的 CUSTOMERS 表

现在对数据库中每个表重复这一过程。记住，要确保每个表至少有一个候选键。

下一步是为每个表建立一个主键。前面已经提到，主键是从可用的候选键中选出来的。当所选择的候选键不止一个时，要注意以下几点。

- 相比复合候选键，应优先选择单字段候选键。
- 如有可能，选取名称中包含表名称的候选键。
- 选择最能代表表主题或对机构内部人员最具代表意义的候选键。

从图 8.12 所示的 EMPLOYEEES 表开始。在评审候选键时发现 Employee Number 比 EmpFirst Name 和 EmpLast Name 的组合更合适，因为迈克的员工已经习惯通过编号识别自己的身份。使用 Employee Number 非常管用，所以选择它作为该表的主键。

Table Structures	
Employees	
Employee Number	CK
Social Security Number	
EmpFirst Name	CCK
EmpLast Name	CCK
EmpStreet Address	
EmpCity	
EmpState	
EmpZipcode	
EmpHome Phone	

图 8.12 迈克自行车行数据库的 EMPLOYEES 表结构

将 Employee Number 指定为正式主键之前，还有最后一个任务：务必确保它是识别特定记录中每个字段值的唯一方式。于是，按照以下步骤测试 Employee Number。

1. 向 EMPLOYEES 表中加载样本数据。
2. 选择一个记录用于测试，记录 Employee Number 的当前值。
3. 检查 Employee Number 后第一个字段的值，考虑下列问题：

这个主键值是识别当前字段值的唯一方式吗？

- a. 是，继续下一字段，重复这一问题。
 - b. 不是，将这个字段从表中清除，继续下一字段，重复这一问题。
(确认是否可以将刚刚清除的字段添加到其他适宜的表结构中，如确实无用，则完全丢弃。)
4. 继续这一过程，直至检查完记录中的每个字段值。

如果某字段中的值不能用 Employee Number 唯一标识，则必须完全清除该字段。不过，Employee Number 确实是被测试记录中所有字段值的唯一识别方式，所以将之定为 EMPLOYEES 表的正式主键，并在该表中其名称旁写

下“PK”字样。然后，对新数据库剩余表重复这一过程，直至每个表都拥有一个主键。

建立每个表的主键时，记住以下规则。

- 每个表有且仅有一个主键。
- 数据库中每个主键必须为唯一的，即任意两个表不得有相同的主键，除非其中一个为子集表。

检查完迈克数据库中的表，你记起来 SERVICES 表是一个子集表。它是在前面阶段创建的（第7章），表示 PRODUCTS 表主题的一个具体的方面。由 Product Name 字段连接 PRODUCTS 表和 SERVICES 子集表。不过，前面已经提到子集表必须与相关联的表共享相同的主键，所以 Product Name（PRODUCTS 表的主键）也是 SERVICES 表的主键。图 8.13 所示为 PRODUCTS 表和 SERVICES 表的结构，其中 Product Name 同为两个表的主键。下一步是为每个表建立主键。

Table Structures			
Products		Services	
Product Number	PK	Product Number	PK
Product Name		Service Type	
Product Description		Materials Charge	
Category		Service Charge	
Wholesale Price		Service Date	
Retail Price			
Quantity On Hand			

图 8.13 建立 SERVICES 子集表的主键

最后一个任务就是对迈克和他的员工展开访谈，评审之前为数据库中表所做的工作。开展访谈时，应注意检查以下几点。

- 数据库中的表表示合适的主题。

- 表名称和描述规范且准确易懂。
- 字段名称规范且准确易懂。
- 每个表中都有适宜字段分配到位。

访谈结束时，大家一致认同表的形式规范，他们所关注的所有主题都收入了数据库中。讨论期间只提出了一个小问题：迈克想添加 Call Priority（呼叫优先级）字段到 VENDORS 表中。鉴于多位供应商能供应同一种产品，迈克想要创建一种方式，指示产品脱销时应优先联系的供应商电话。于是，将这个新字段添加到 VENDORS 表中，并将访谈引向结束。

小结

本章开篇即讨论了键的重要性。键有多种，每一种都在数据库中发挥着不同的作用。每个键都有着特定的功能，比如承担识别记录的唯一方式，建立各种完整性以及建立表之间的关系。为每个表建立合适的键，就能确保完善的表结构。

接着，讨论了为每个表建立键的过程。一开始就确认了四种类型的键：候选键、主键、外键和非键。首先，论述了为每个表建立候选键的过程。其中提到了候选键的要素和如何确保字段（或字段组）满足这些要素。在表中无候选键或新字段比现有候选键字段更为有效的情况下，可以创建和使用人造候选键。

然后，又继续讨论了主键。主键是从候选键中选择出来的，必须符合主键的各个要素。文中介绍了一套指南，帮助读者判断候选键是否能成为主键。随后，还谈到了如何确保所选的主键为特定记录及其系列字段值的唯一识别方式。当某主键不是特定字段值的唯一识别方式时，必须将该字段从该表中移除，确保该表结构的完整性。另外，每个表有且仅有一个主键。

对于剩余的候选键，应将它们指定为替换键。当数据库在 RDBMS 程序中实施时，这些键将非常有用，因为它们能提供识别特定记录的替代方式。非键是非候选键、主键、替换键或外键的字段。非键字段表示表主题的特征，

主键是识别其值的唯一方式。

表层次完整性是接下来讨论的一个主题，通过使用主键建立表层次完整性，受主键元素约束。

本章以引导进一步的用户和管理人员访谈结尾。通过这些访谈，评审之前对表所做的工作，验证和确认当前数据库结构。

思考题

1. 说出键之所以重要的三个原因。
2. 四种主要类型的键分别是什么？
3. 候选键的用途是什么？
4. 说出候选键的四个要素。
5. 判断题：候选键可由多个字段组成。
6. 一个表能拥有多个候选键吗？
7. 什么是人造候选键？
8. 表中最重要键是什么？
9. 这个键为什么重要？
10. 如何建立一个主键？
11. 说出主键的四个要素。
12. 选定主键之前，还要做些什么？
13. 什么是替换键？
14. 建立表层次完整性能确保什么？
15. 为什么要评审初始表结构？



第9章

字段说明

长期以来，我一直记着一句格言：细节至关重要。

——夏洛克·福尔摩斯

《福尔摩斯冒险史》

本章内容

字段说明为何重要

字段级完整性

字段说明之剖析

使用独特、通用和可复制的字段说明

定义每个字段的字段说明

案例分析

小结

思考题

字段是数据库的基石。它们表示的是机构重要主题的特征。字段存储了被机构用作基本信息的数据，这些信息对于机构的日常运转、成功以及未来的发展都至为重要。尽管拥有固有价值，但是字段仍然是最容易被忽视、价值亟待充分开发的机构财富。通常，人们少有时间或根本没有时间用在确保数据库中字段的结构和逻辑完整性上。

谈及数据完整性，人们总是长篇累牍、滔滔不绝，但是言而不行、少以实际行动。许多人认为留意数据输入人员，数据库拥有安全的用户界面，就能极大地减少与数据相关的潜在问题。这种治标不治本的解决方法通常源于对数据完整性的错误认识，认为建立适当的数据完整性需要花费大量时间。然而，值得注意的是，人们不在建立数据完整性上投入时间，通常就需要花费大量时间修正设计不当的数据库，而修正数据库所消耗的时间一般为之前规范设计数据库的三倍。

在本章中，读者将学会如何通过为数据库中每个字段定义**字段说明**，从而建立数据完整性。首先，读者将学习字段说明所包含的三个系列的元素；然后，就是如何与用户和管理人员访谈，帮助定义字段说明。

字段说明为何重要

不管传言如何，为数据库中每个字段建立字段说明所花费的时间都是一笔可观的投资，有利于建立统一的数据和质量信息，**绝不是**浪费时间。事实上，如果对此敷衍了事或断然放弃，最后只会浪费**更多**时间。逃避这一责任意味着必然会遭遇（并忍受）错误的数据和不准确的信息。

下列为字段说明之所以至关重要的几个原因。

- **字段说明有助于建立和执行字段级完整性。**落实这些说明能确保每个字段中数据一致且有效。
- **为每个字段定义字段说明能提升整体数据完整性。**记住，字段级完整性是数据完整性的四大部分之一。字段级完整性（在一定程度上）能提升之前阶段所建立的表层次完整性。（当了解字段说明的逻辑要素时，这一点就会显现出来。）
- **定义字段说明能获得对数据的性质和用途的完整认识。**了解数据意味着可以判断数据是否对机构真正必要和重要，可以学会如何使之得到最充分利用。
- **字段说明构成数据库的“数据字典”。**每个字段说明存储了数据库中特定字段的特征方面的数据。为数据库中所有字段建立的整套说明构

成了解读数据库结构的字典。这一数据字典在 RDBMS 程序中实现数据库时尤其有用，可以将之当作创建字段和设置字段基本属性的指南。这些说明也有助于决定在数据库的用户界面程序中，需要执行哪些数据输入和数据验证程序。

记住，数据库中数据一致性、质量和准确性的水平（以及从中检索得出的信息）都与这些说明的完善程度成正比。如果机构非常依赖从数据库中检索的信息，就必须为每个字段建立完善的字段说明。

字段级完整性

整套字段说明定义完成后，字段就能达成**字段级完整性**。字段级完整性能保证以下优势。

- 字段特性和用途明确，它所出现的表都得到正确识别。
- 整个数据库中字段定义一致。
- 字段的值一致且有效。
- 修改、比较和操作字段值所运用的方法有了明确界定。

有了一整套字段说明，兼之字段完全满足理想字段的要素，就能确保字段结构完善并得到优化设计。实际上，字段满足理想字段的要素，会使定义一套说明变得相对容易。

如果对特定字段是否完全满足理想字段要素仍存有疑虑，现在就是再次评审该字段的好时机。如果认定它不满足，就运用相应技巧解决这一问题，并对改变进行适当的调整；否则，可以开始为每个字段定义字段说明。为了便捷，再次列出理想字段的要素。

- 代表表主题的鲜明特征。
- 仅包含一个值。
- 无法分解为更小的元素。
- 不含计算值或串联值。
- 在整个数据库结构中独一无二。

- 主要特性始终保持不变。

字段说明之剖析

字段说明包含定义每种字段属性的各个元素。说明中所有元素分为一般元素、物理元素和逻辑元素。这些要素分类在定义说明时，有助于集中于字段的某个具体方面，也能提供一种轻易就能找到特定要素的方式。

下列为每个类别中的元素。

- 一般元素 (general elements): 字段名称 (field name)、父表 (parent table)、标签 (label)、说明类型 (specification type)、源说明 (source specification)、共享 (shared by)、别名 (alias)、描述 (description)
- 物理元素 (physical elements): 数据类型 (data type)、长度 (length)、小数位 (decimal places)、字符支持 (character support)、输入掩码 (input mask)、显示格式 (display format)
- 逻辑元素 (logical elements): 键类型 (key type)、键结构 (key structure)、单值性 (uniqueness)、null 支持 (null support)、值的输入者 (values entered by)、要求值 (required value)、默认值 (default value)、值的范围 (range of values)、编辑规则 (edit rule)、允许的比较 (comparisons allowed)、允许的运算 (operations allowed)

图 9.1 所示为字段说明表的示例。在后面的部分，涉及字段说明的图示都是以这一份字段说明表作为基础。

FIELD SPECIFICATIONS							
General Elements							
Field Name:	Specification Type: <input type="checkbox"/> Unique <input type="checkbox"/> Generic <input type="checkbox"/> Replica						
Parent Table:	Source Specification:						
Label:							
Shared By:							
Alias(es):							
Description:							
Physical Elements							
Data Type:	Character Support:						
Length:	<input type="checkbox"/> Letters (A-Z) <input type="checkbox"/> Keyboard (., / \$ # %)						
Decimal Places:	<input type="checkbox"/> Numbers (0-9) <input type="checkbox"/> Special (© ® ™ Σ π)						
Input Mask:							
Display Format:							
Logical Elements							
Key Type:	<input type="checkbox"/> Non	<input type="checkbox"/> Primary	Edit Rule: <input type="checkbox"/> Enter Now, Edits Allowed <input type="checkbox"/> Enter Now, Edits Not Allowed <input type="checkbox"/> Enter Later, Edits Allowed <input type="checkbox"/> Enter Later, Edits Not Allowed <input type="checkbox"/> Not Determined At This Time				
	<input type="checkbox"/> Foreign	<input type="checkbox"/> Alternate					
Key Structure:	<input type="checkbox"/> Simple	<input type="checkbox"/> Composite					
Uniqueness:	<input type="checkbox"/> Non-unique	<input type="checkbox"/> Unique					
Null Support:	<input type="checkbox"/> Nulls Allowed	<input type="checkbox"/> No Nulls					
Values Entered By:	<input type="checkbox"/> User	<input type="checkbox"/> System					
Required Value:	<input type="checkbox"/> No	<input type="checkbox"/> Yes					
Default Value:							
Range of Values:							
Comparisons Allowed:							
<input type="checkbox"/> Same Field	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/>	<input type="checkbox"/> <	<input type="checkbox"/> <=
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/>	<input type="checkbox"/> <	<input type="checkbox"/> <=
<input type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/>	<input type="checkbox"/> <	<input type="checkbox"/> <=
Operations Allowed:							
<input type="checkbox"/> Same Field	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation	
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation	
<input type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation	

图 9.1 字段说明表

一般元素

一般元素类别下的项表示字段的最基本属性。它们提供的是字段用途、该字段所在表的名称，以及特定情况下该字段所使用的假名这方面的信息。

字段名称

字段名称是整个数据库中识别特定字段的唯一方式，而且所用词语也是绝对最少的。在之前的过程中（参见第7章“建立表结构”）已经创建和定义了字段名称，所以可以直接使用。

父表

表结构中包含给定字段的表成为该字段的父表。这个表是该字段唯一出现的表，除非该字段参与到关系的建立中。（这一例外详见第10章“表关系”。）例如，STUDENTS 是 StudFirst Name 字段的父表。

标签

标签是一个替代名，通常借以识别数据库的最终用户应用程序界面中的该字段。例如，可将 Qty On Hand 用作 Quantity On Hand 字段的标签，因为机构内部许多人早已习惯这一名称。如果想在数据输入表单上节省空间或在特定报表中容纳更多的字段，标签尤其管用。

应避免在表结构中使用标签作为正式字段名称；否则，一些人可能因此误解或误判该字段。正式字段名称应始终采用最精确的名称，最终用户应用程序界面中则可（谨慎）使用标签。这将确保你能将这两项区分清楚。

说明类型

给定字段所设定的元素取决于定义该字段的说明类型。定义说明有三种方式。

1. **独特**：这是所有字段的默认说明，充当其他字段模板的字段和参与表关系中作为外键的字段除外。这种说明可包含除源说明元素外所

有元素，而且元素设置将仅适用于字段名称所指示的字段。

2. **通用**：这一说明作为其他字段说明的模板，有助于确保字段定义始终一致，意义普遍通用。例如，可为通用字段 `State` 创建这一说明，然后把它当作数据库中所有其他 `State` 字段的基础。诸如 `CustState`、`EmpState` 和 `VendState` 的字段具有相同的意义（都表示美国的州），不过它们之间有明显的区别，都是相互独立的字段。（如果你还记得，第 6 章“分析现有数据库”介绍开发初始字段列表和第 7 章理想字段要素，都曾提到通用字段。）

通用说明要求使用广为熟知的非特殊字段名称和元素设置。不过，除开父表、标签、共享、别名和源说明，其他元素都可包含在内。

3. **可复制**：这一默认说明适用于基于通用字段的字段和充当表关系中外键的字段，它从既有说明中借鉴了大部分元素设置。可以收纳源说明中未包含的元素，也可更改从源说明中汲取的任意元素设置。

本章“使用独特、通用和可复制的字段说明”一节将介绍如何定义每种说明。

源说明

这一元素仅用于可复制说明，它显示了当前说明所基于的特定字段说明的名称。（下一节中展示了这一元素的良好示例。）

共享

这一元素显示了其他共享这一字段的表名称。此处出现的表名称仅仅是与该字段父表有明确关联的。例如，一个数据表 `EMPLOYEES` 与两个子集表 `PART-TIME EMPLOYEES` 和 `FULL-TIME EMPLOYEES` 通过 `Employee ID Number` 字段建立关联，就会使用“`PART-TIME EMPLOYEES`，`FULL-TIME EMPLOYEES`”作为这一元素的设置。

别名

别名是偶尔用于代表字段的一个或一组名称。比如，相同表中该字段必

须出现两次，就会使用别名。不妨假设，某机构习惯于通过 Employee ID Number 字段中的唯一值识别员工。再来考虑一下图 9.2 中的 SUBSIDIARIES 表结构（仅为该表的部分）。

Table Structures	
<u>Subsidiaries</u>	
Subsidiary ID Number	
Subsidiary Name	
Employee ID Number	
Employee ID Number	
SubsStreet Address	
SubsCity	

图 9.2 含两个相同字段的表

在本例中，每个子公司都分别有一个董事长和副董事长。因为他们在子公司中的地位，他们两人都必须出现在表中，所以表结构中有两个 Employee ID Number 字段。然而，规范的数据库设计规定表中这个字段只能出现一次；显然，这里出现了问题。唯一的解决方案是使用别名代替 Employee ID Number 字段。例如，（为表述明确）可使用 President ID 代替第一个 Employee ID Number，Vice President ID 代替第二个 Employee ID Number。有了这些别名，这两个员工就都有了合适的指称。图 9.3 所示为修正后的结构。

Table Structures	
<u>Subsidiaries</u>	
Subsidiary ID Number	
Subsidiary Name	
President ID Number	
Vice President ID Number	
SubsStreet Address	
SubsCity	

图 9.3 使用别名代替 Employee ID Number 字段

尽管在这些情况中使用别名是可行的，但是应对此保持谨慎；否则，将难以管理和维护，最终隐没或掩盖原始字段的真正含义，造成对该数据实际意义的误解。当着手建立表关系时，这个问题将变得更加明显。

描述

描述是对字段的完整解读。编辑字段描述让人受益匪浅，因为它让你（以及机构内部所有人）认真思考将存储在该字段中数据的性质。如果无法编写出合适的描述，就能相对准确地认定该字段需要进一步改进。

在之前的阶段中介绍了编辑表描述的一整套指南。同样，这里也有一套指南，规定了如何制定合适的字段描述。

编写字段说明的指南

- **准确描述该字段，阐明其用途。**描述应定义该字段所表示的含义，对字段名称进行补充说明。它也应该阐明该字段在表中的作用或其与表主题的关系。下列为示例：

CustCity——顾客住所或经营业务所在城市。这是顾客完整地
址的一个重要部分。

- **陈述简明扼要。**描述应避免出现含糊不清的语句或词语。尽管描述应尽可能完善，但是也应使用尽可能少的描述，传达出必要信息。正如表描述中所提到，烦琐的语言让人难以理解。
- **避免重申或改述使用字段名称。**这些做法都无益于阐明该字的特性和用途。记住，描述的目的是提供对该字段的完整解读。下列为不当描述的示例：

CustLast Name——顾客的姓氏。

按照下列做法，描述将更为有用：

CustLast Name——顾客的父姓或夫姓，用于正式场合中指代该顾客。

- 避免使用专业术语、缩略语和缩写。尽管机构内部有一部分人能理解这些用语，但使用人人都懂的术语更为妥当。记住，描述必须让人一读即懂。例如，下列陈述应予避免：

Employee ID Number——用于识别机构员工的独有编号。它是 SSP 的组成部分。

这一描述的问题是无法从字面确定缩略语 SSP 的意思。可以将完整的术语拼写出来，解决这一问题。不过，最好重新叙述字段的用途。

- 切勿包含具体实施信息。不必提及该字段出现在特定数据输入界面或在具体程序编码中使用的信息。这种信息更合适在数据库整体开发的实施阶段出现。
- 此描述不得依赖于另一字段描述。每个描述都应尽可能完善，并独立于数据库中其他所有描述。相互依赖的描述会造成不必要的混淆，可能不慎掩盖该字段的真实特性和用途。避免使用如下描述：

Item Reorder Level——特定产品必须存在的最少数量。（参见 Quantity On Hand 的描述。）

- 切勿使用示例。第 7 章中已经提到，描述中应避免使用示例，因为示例需要依靠附加信息才能传达完整的意思。彻底禁用示例，就能确保描述简洁明了。

图 9.4 所示为字段 Employee ID Number 的字段说明中一般元素的部分。

General Elements		
Field Name:	Employee ID Number	SpecificationType: <input checked="" type="checkbox"/> Unique <input type="checkbox"/> Generic <input type="checkbox"/> Replica
ParentTable:	Employees	Source
Label:	Employee #	Specification:
Shared By:	Full-Time Employees, PartTime Employees, Customers	
Alias(es):		
Description:	A unique number used to identify each employee within our organization. It is assigned during the first day of Employee Orientation and remains with the employee throughout the duration of his or her employment.	

图 9.4 Employee ID Number 字段一般元素

物理元素

这一范畴与字段结构相关。其中的元素采用通用术语表示，因为每种 RDBMS 程序实施它们的方式都有少许区别。在这一阶段建立这些元素，有助于确保整个数据库中字段定义统一，也能减少在 RDBMS 程序中实现字段结构所花费的时间。

数据类型

这一元素显示的是该字段中所存储数据的性质。

第 1 章“关系数据库”即提到，结构化查询语言（或 SQL）是用于创建、修改、维护和查询关系数据库的标准语言。SQL 实际上是由美国国家标准协会（ANSI）和国际标准化组织（ISO）共同提出的完全文档化标准。尽管该标准的当前版本是 SQL 2011，但是大多数主流 RDBMS 程序仍然支持的是之前版本的 SQL，比如 SQL/92 和 SQL 2008。

SQL 标准定义了八种主要数据类型，每种数据类型都有一个或多个独特名称。下列为每种数据类型的简单定义。

1. **字符**：这种数据类型存储的是固定长度或可变长度字符串，字符串包含一个或多个可打印字符。固定长度字符的数据类型称为 CHARACTER 或 CHAR，可变长度字符的数据类型称为 CHARACTER VARYING、CHAR VARYING 或 VARCHAR。

2. **国际字符**：这种数据类型与字符数据类型相同，不过它也能存储外语字符集。固定长度的国际字符的数据类型称为 NATIONAL CHARACTER、NATIONAL CHAR 或 NCHAR，可变长度的国际字符的数据类型称为 NATIONAL CHARACTER VARYING、NATIONAL CHAR VARYING 或 NCHAR VARYING。

3. **二进制**：这种数据类型存储二进制数据，比如图像、声音、视频或复杂嵌入式文档，包括文档、电子表格等。这种数据类型通常被称为 BIT 或 BIT VARYING。

4. **精确数字**：这种数据类型存储整数和具有小数位的数。大多数 RDBMS 程序执行的精确数位分别为 NUMERIC、DECIMAL (DEC)、INTEGER (INT)、BIGINT 和 SMALLINT，每一名称都对应该字段受值的范围。

5. **近似数字**：这种数据类型存储小数和指数。大多数 RDBMS 程序执行的近似数位分别为 FLOAT、REAL 和 DOUBLE PRECISION，每一名称都对应该字段受值的范围。

6. **布尔值**：这种数据类型存储真值和假值，通常为一个二进制位。一些 RDBMS 程序使用 BIT、INT 或 TINYINT 存储这种数据类型。

7. **日期和时间**：在多数 RDBMS 程序中这种数据类型通常被称为 DATE、TIME 或 TIMESTAMP，它存储日期、时间和两者的结合。注意，各种 RDBMS 程序执行这一数据类型的方式不一，所以务必参考所选 RDBMS 程序资料，确认该 RDBMS 程序如何处理日期和时间。

8. **间隔**：这种数据类型存储两个创建时间值之间的时间量，表示为年、月、年/月/日、时间或日/时间。并非所有数据库系统都支持这种数据类型，所以要检查所选 RDBMS 程序资料获取进一步信息。

许多 RDBMS 程序提供上述数据类型之外的附加数据类型，被称为**扩展数据类型**。扩展数据类型的示例包括 MONEY/CURRENCY 和 SERIAL/ROWID（作为独特的行标识符）。

本书之所以展示 SQL 标准数据类型，是因为将在 RDBMS 程序的实际操作中遇到它们（或其变体）。不过，本书并未提供它们的过多细节，因为它们在各种 RDBMS 程序中有所不同；必须参考所选 RDBMS 资料，确定该 RDBMS 所支持的数据类型以及如何实施。

本来，可以使用任意 SQL 数据类型（除布尔值和间隔外）作为给定说明中数据类型元素的设置。然而，由于数据类型实施的不一致，本书推荐使用下列三种一般数据类型中的一种作为这一元素的设置。

1. **字母数字**：这种数据类型可存储任何字母、数字、键盘字符或特殊

字符的组合。键盘字符包括逗号、美元符号、感叹号、百分比符号和句号。特殊字符包括版权符号、商标符号和圆周率符号。

2. **数字**：这种数据类型只能存储整数和实数。它不接受具有前导零的数字（例如 0000234），因为这些数字不是真正的数字。

3. **日期和时间**：这种数据类型存储日期、时间或两者的结合。

这些数据类型用于指示该字段所存储数据的性质非常合适，也的确让用户和管理人员更容易理解。使用一般数据类型有助于避免不必要的误解，尤其是在与用户和管理人员评审说明的时候。

❖ **注意**：本书后续部分将使用这些一般数据类型作为基础，所有数据类型引用和示例都基于此。在特定 RDBMS 程序实施数据库时，必定会对此做出相应的调整。

长度

这个元素指定用户可以输入任意字段值的字符总量。实现数据库所使用的 RDBMS 程序决定了为这一元素设置的字符最大数量。尽管理论上可以为任意数据类型设置长度元素，但是一些 RDBMS 程序不允许为数字字段指定长度。相反，RDBMS 程序根据该字段所存储数字类型设置数字字段长度，比如整数、长整数或实数。

小数位

小数位指示实数小数点右边的位数。位数决定了实数的精确度。例如，许多机构都要求所有的货币值都精确到小数点右边第四位。

字符支持

这一元素显示的是用户输入给定字段值的字符类型。设置和执行这一元素有助于避免该用户引入无用数据，从而提升字段级完整性。

不妨假设正在处理字段 CustState，其数据类型为字母数字。这种数据类型适合该字段，因为它允许用户输入的字段值中包含字母。但是，它也允许

使用数字、键盘字符和扩展字符，这意味着该用户可以输入无用的值，因为所有州名或州名缩写中都不包含字母外的其他字符。使用字符支持元素定义该用户可输入字段值中的字符，就能解决这个问题。（这一方法在“逻辑元素”一节中解决了一个字母的有效组合。）

可以选择包含或排除下列字符。

- **字母：**英文字母表所有字母，包括é和ñ之类字母。
- **数字：**从0至9。
- **键盘字符：**除字母和数字之外的所有标准字符，比如星号、和号、括号、插入符号、逗号、等号、感叹号、圆括号、百分号、句号、英镑符号、问号、引号、分号、斜线或竖线。注意，字段说明表包含了属于这一类别的字符。
- **特殊字符：**只有通过标准键和 CTRL、ALT 及（或）SHIFT 键的特殊组合，或者在特殊软件的帮助才能产生的字符。这一类别的字符包括复杂数学符号、版权符号、分数符号、圆周率符号以及商标符号。字段说明表也包含了这一类别的字符。

输入掩码

这一元素指示的是用户应采取的输入该字段数据的方式。例如，输入日期的方式多种多样，比如，“01/01/12”、“01-01-12”和“01-Jan-2012”。使用输入掩码能确保用户采用统一的输入方式，（在这一示例中）防止对日期顺序的混淆。

各种 RDBMS 程序实施的输入掩码方式也不尽相同，所以应该对这一元素使用相对通用的设置。（如适宜，可指定多重输入掩码。）例如，使用“mm/dd/yy”作为日期字段的输入掩码。这一掩码表明了日期顺序（月，日，年）、日期结构（每部分都采用两位数，例如 05/16/12）以及日期分隔符（斜线）。

显示格式

这一元素规定了字段显示在界面中或文档中打印出来时，它的值的外

观。通过显示格式的方式展示字段值比该字段值的输入方式更有意义、更具可读性。例如，输入给定日期的方式是“03/13/2012”，但是“2012 年 3 月 13 日”更为简单明了。

对这一元素使用通用设置，正如处理输入掩码一样；各种 RDBMS 程序实施的显示格式也是各有特色。例如，Date Hired 字段可以使用“月日，年”作为显示格式。同时，也可以使用一个完整的句子，比如下列示例为 Company Name 字段的显示格式设置：

每个单词都应以大写字母开头。

图 9.5 所示为 Employee ID Number 字段说明的物理元素部分。

Physical Elements		Character Support:	
Data Type:	Numeric	<input type="checkbox"/> Letters (A-Z)	<input type="checkbox"/> Keyboard (. , / \$ # %)
Length:	4	<input checked="" type="checkbox"/> Numbers (0-9)	<input type="checkbox"/> Special (© ® ™ Σ π)
Decimal Places:	0		
Input Mask:	####		
Display Format:	0000		

图 9.5 Employee ID Number 字段的物理元素类别

逻辑元素

这一类别主要与字段中的值有关。其中的元素涉及问题诸如每个值是否应为唯一值，何时应输入一个值，是否可以编辑某个值，以及每个值可以进行的比较和操作类型等。设置这些元素有助于建立和实施大部分的字段级完整性。

键类型

这一元素指定的是一个字段在表中的作用，在为该表建立主键时进行认定。如你所知，字段可以成为非键、主键或替换键。第 10 章中将介绍外键的详情以及在字段说明表中合适指定外键。

键结构

这一元素指示一个字段被指定为主键时，它是作为单字段主键还是复合主键的一部分。

单值性

这一元素表明一个字段的值是否为唯一值。当键类型元素设置为“primary（主键）”，这一元素设置“unique（唯一）”；否则，通常将这一元素设置为“non-unique（非唯一）”。

处理非键字段时，先考虑它的值将作何用途，再决定是否为唯一值。不妨来看看图 9.6 中的 DEPARTMENTS 表结构。

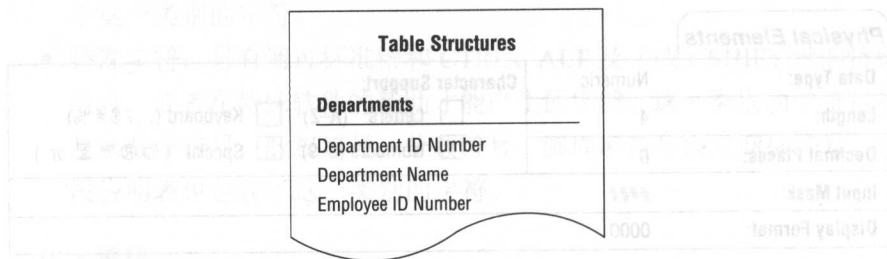


图 9.6 Employee ID Number 的值是否应为唯一值

在这个例子中，Employee ID Number 字段识别的是负责管理特定部门的人。假设，一个人只能管理一个部门，这个字段的值就是唯一值；因此，应该将单值性元素设置为“唯一”。

null 支持

这一元素指明一个字段是否接受 null 值。它通常设置为“禁用 null”，在字段作为主键或替换键，或者字段的要求值元素设置为“是”的情况下尤为如此。然而，如果某字段能接受 null 值，其理由充分，就可以设置为“允许 null”。例如，CustCounty 字段必须接受 null，因为顾客可能不清楚他居住的县名。（当然，一旦他补充了这一信息，就不再是 null。）

记住，null 不表示一个空格，而表示一个缺失值或未知值。用户通常误

将空格用来表示无用值，比如“无”、“不适用”、“无响应”和“不缺”。如果这些值对特定字段有效，就要确保将它们包含在该字段的值的范围元素中。最重要的是，慎用 `null`，禁用空格。

值的输入者

这一元素指明了字段值的来源。输入字段值的方式分为用户手动输入和数据库应用程序自动输入；唯有程序的开发者提供生成字段值的方式，应用程序才能为该字段提供值。注意，代表该数据库应用程序的设置为“系统”。

要求值

这一元素指明是否要求用户为某字段输入值。尽管表中大多数字段通常都将这一元素设置为“否”，但是当该字段作为主键时必须设置为“是”。再比如，`CustZipcode` 字段表示向给定顾客发送的信件或包裹必须包含邮政编码，以便邮局服务处准确处理。对于此类字段，可能也需要将要求值设置为“是”。

默认值

这是在不存在更合适的值且不可用 `null` 的情况下，用户可以输入字段的一个值。使用默认值应非常谨慎，唯有默认值有意义时才可用。例如，大部分顾客都居住在华盛顿，`CustState` 字段就可以选择“WA（华盛顿英文单词的前两个字母）”作为默认值。相反，`Date Hired` 字段不宜选择“01/01/12”作为默认值，因为它只是任意值，实际意义不大。

值的范围

这一元素指示的是某一字段所有可能的有效值。设置这一元素的方式多种多样，比如使用下限和上限（1,000 至 9,999）或者一系列具体的值（“WA”，“OR”，“ID”，“MT”）。下列为建立值的范围可遵行的三个类别。

1. 一般——这一字段的所有可能值的完整集合。例如，`CustState` 字段的一般值的范围可包括美国所有州的正确缩写。

2. 视完整性而定——基于该字段在表关系中作用的值的集合。（详情参见第10章。）
3. 视业务而定——根据特定业务要求产生的值的集合。机构通常对限制字段的值的范围有各种要求。例如，一家机构业务范围严格限制在太平洋西北地区，CustState 字段的有效值的范围就是“WA”、“OR”、“ID”和“MT”。（详情见第11章“业务规则”。）

在这一阶段，只关注一般值的范围。等到建立表关系和业务规则时，将再次考虑值的范围元素。

值得注意的是，应避免将“其他”和“杂项”两个值设置在任何值的范围类别中。这两个值既不明确，也无意义。其出现可视为懒惰心理作祟，也表明需要对该字段进行评审，做进一步改进。禁用这些值能避免造成不必要的误解和潜在问题。

编辑规则

这一元素指明用户何时可以向字段输入值以及是否可以修改该值。它可设置为以下四种选项。

1. **立即输入，可编辑**：用户在该字段父表中创建新记录时，必须输入该字段的值。以后随时都可对该值进行编辑。
2. **随后输入，可编辑**：用户在该字段父表中创建新记录时，可适时输入该字段的值。这并不意味着该字段可永久为 null；该用户必须在随后的某一时间向这一字段输入值。输入值后，可随时对其进行编辑。
3. **立即输入，不可编辑**：用户在该字段父表中创建新记录时，必须输入该字段的值，但过后不可编辑。
4. **随后输入，不可编辑**：用户在该字段父表中创建新记录时，可适时输入该字段的值。这并不意味着该字段可永久为 null；该用户必须在随后的某一时间向这一字段输入值。输入值后，不可编辑。

将编辑规则元素设置为第二或第四选项时，应使用默认值；这样就能防

止该用户当前不输入值，导致该字段值为 null。

允许的比较

这一元素指明用户检索某字段信息时，可以对该字段值运用的比较类型。比较类型分六种：等于 (=)、不等于 (\neq)、大于 (>)、小于 (<)、不小于 (\geq) 以及不大于 (\leq)。这个元素也指示用户是否可以将下列项与给定字段值相比较。

- 相同字段中的其他值。当一个字段充当主键时，这一选项适用于相关外键字段的值。（详情参见下一章。）
- 数据库中另一表或父表中另一字段的值。
- 值表达式，即某种涉及字段值、文字值或两者结合的运算。它输出的单一值可以与字段值进行比较：(Retail Price - 2.50) 就是一个值表达式的示例。

限制用户可运用的字段值比较类型，就能避免用户进行无意义的比较。不妨假设，某用户处理 Employee ID Number 字段，该字段基于数值数据类型。除非指明其他方式，否则他就能做以下比较：

员工表中某员工号是否不小于兼职员工表中某员工号？

尽管在数值字段中“不小于”比较一般是可接受的，但在这个例子中却不合适；这种比较根本就无意义。

同样，对给定 Employee ID Number 值与 EMPLOYEES 表或数据库另一表中其他数值字段的值进行比较也毫无意义；因此，以下比较无效：

员工表中某员工号是否不小于产品表中现存数量？

不过，比较 EMPLOYEES 表中给定 Employee ID Number 值与相关数据表或相关子集表中另一 Employee ID Number 值则合情合理。所以，下列比较有效：

员工表中某员工号是否等于兼职表中某员工号？

下面将举例说明用户可以比较完全不同的两个字段的值。例如，对 Date Shipped 字段和 Date Ordered 字段进行比较，完全符合情理：

发货日期的当前值是否不小于订单日期的当前值？

幸运的是，这一比较是可行的，因为用户当然不希望看见发货日期的值比订单日期的值小！

当为特定字段设置允许的比较这一元素时，先考虑将如何使用该字段值，从而指定合适的比较类型。等到建立表关系和定义业务规则时，极有可能再回顾这一元素。

允许的运算

这一元素指定用户可对该字段值进行的运算类型。运算类型分五种：加（+）、减（-）、乘（×）、除（÷）以及串联。（显然，这些运算的任意组合同样有效。）这个元素也指示一次运算是否可以包含：

- 相同字段中的另一个值。
- 父表或数据库中其他表中另一字段的值。
- 值表达式的结果（某种涉及字段值、文字值或两者结合的运算，输出单一值）。

限制用户对字段值可进行的运算类型，就能避免用户定义无意义的运算。再来考虑一下 Employee ID Number、Date Shipped 和 Date Ordered 字段。用户对 EMPLOYEES 表中两个 Employee ID Number 值进行数学运算毫无意义，对 Employee ID Number 值和另外一个数值字段的值之间的此类运算效果也是如此。然而，就 Date Shipped 字段而言，给定 Date Shipped 值和数据库中其他适宜日期字段值之间的运算却是恰当的。例如，用户用 Date Shipped 值减去相应 Date Ordered 值，就能确认该顾客下订单日期与发货日期之间的时间差。

当为特定字段设置允许的运算这一元素时，先考虑将如何使用该字段值，从而指定合适的运算类型。等到建立表关联和定义业务规则时，极有可

能再回顾这一元素。

图 9.7 所示为 Employee ID Number 字段说明表的逻辑元素部分。

Logical Elements									
Key Type:		<input type="checkbox"/> Non	<input checked="" type="checkbox"/> Primary	Edit Rule:					
		<input type="checkbox"/> Foreign	<input type="checkbox"/> Alternate	<input type="checkbox"/> Enter Now, Edits Allowed					
Key Structure:		<input checked="" type="checkbox"/> Simple	<input type="checkbox"/> Composite	<input checked="" type="checkbox"/> Enter Now, Edits Not Allowed					
Uniqueness:		<input type="checkbox"/> Non-unique	<input checked="" type="checkbox"/> Unique	<input type="checkbox"/> Enter Later, Edits Allowed					
Null Support:		<input type="checkbox"/> Nulls Allowed	<input checked="" type="checkbox"/> No Nulls	<input type="checkbox"/> Enter Later, Edits Not Allowed					
Values Entered By:		<input type="checkbox"/> User	<input checked="" type="checkbox"/> System	<input type="checkbox"/> Not Determined At This Time					
Required Value:		<input type="checkbox"/> No	<input checked="" type="checkbox"/> Yes						
Default Value:									
Range of Values:		1000-9999							
Comparisons Allowed:									
<input checked="" type="checkbox"/> Same Field	<input type="checkbox"/> All	<input checked="" type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/> ≠	<input type="checkbox"/> <	<input type="checkbox"/> <=		
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/> ≠	<input type="checkbox"/> <	<input type="checkbox"/> <=		
<input checked="" type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input checked="" type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/> ≠	<input type="checkbox"/> <	<input type="checkbox"/> <=		
Operations Allowed:									
<input type="checkbox"/> Same Field	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation			
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation			
<input type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation			

图 9.7 Employee ID Number 字段逻辑元素类别

使用独特、通用和可复制的字段说明

本章前面部分已经提到定义独特、通用和可复制的字段说明。遵循以下指南，就能确保定义正确的字段说明。

- 在整个数据库中仅出现一次的字段或主键字段可使用独特字段说明。
- 在整个数据库中作为其他字段模板的字段可使用通用字段说明。记住，须使用非特殊字段名称以及普遍通用的元素设置。
- 基于给定通用字段的字段或作为表关系中外键的字段可使用可复制的字段说明。

图 9.8 所示为 Vendor ID Number 字段完整的独特字段说明。

FIELD SPECIFICATIONS			
General Elements			
Field Name:	Vendor ID Number	Specification Type:	<input checked="" type="checkbox"/> Unique <input type="checkbox"/> Generic <input type="checkbox"/> Replica
Parent Table:	Vendors	Source Specification:	
Label:	Vendor #		
Shared By:	Products		
Alias(es):			
Description:	A unique number used to identify each Vendor that supplies our organization with goods or services. It is assigned when we place the first order for such goods or services with the Vendor.		
Physical Elements			
Data Type:	Numeric	Character Support:	
Length:	6	<input type="checkbox"/> Letters (A-Z)	<input type="checkbox"/> Keyboard (., / \$ # %)
Decimal Places:	0	<input checked="" type="checkbox"/> Numbers (0-9)	<input type="checkbox"/> Special (© ® ™ Σ π)
Input Mask:	#####		
Display Format:	000000		
Logical Elements			
Key Type:	<input type="checkbox"/> Non <input checked="" type="checkbox"/> Primary	Edit Rule:	<input type="checkbox"/> Enter Now, Edits Allowed
	<input type="checkbox"/> Foreign <input type="checkbox"/> Alternate		<input checked="" type="checkbox"/> Enter Now, Edits Not Allowed
Key Structure:	<input checked="" type="checkbox"/> Simple <input type="checkbox"/> Composite		<input type="checkbox"/> Enter Later, Edits Allowed
Uniqueness:	<input type="checkbox"/> Non-unique <input checked="" type="checkbox"/> Unique		<input type="checkbox"/> Enter Later, Edits Not Allowed
Null Support:	<input type="checkbox"/> Nulls Allowed <input checked="" type="checkbox"/> No Nulls		<input type="checkbox"/> Not Determined At This Time
Values Entered By:	<input type="checkbox"/> User <input checked="" type="checkbox"/> System		
Required Value:	<input type="checkbox"/> No <input checked="" type="checkbox"/> Yes		
Default Value:			
Range of Values:	100000-200000		
Comparisons Allowed:			
<input checked="" type="checkbox"/> Same Field	<input type="checkbox"/> All	<input checked="" type="checkbox"/> =	<input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
<input checked="" type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input checked="" type="checkbox"/> =	<input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
Operations Allowed:			
<input type="checkbox"/> Same Field	<input type="checkbox"/> All	<input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation
<input type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation

图 9.8 Vendor ID Number 字段独特字段说明

以下为这一说明需要注意的几点。

1. 通用元素“Shared By (共享)”表明, 这个字段也出现在 **PRODUCTS** 表中。这一点合情合理, 因为每种产品都必定与特定的供应商相关。(详情见下一章。)
2. 检查逻辑元素单值性、null 支持、要求值以及编辑规则的设置。这样设置是因为键类型元素设置为“Primary”。实际上, 应该对所有主键字段使用这些元素设置。
3. 逻辑元素允许的比较设置为“Same Field (相同字段)—Equals (等于)”, 这样用户就能比较 **VENDORS** 表中 Vendor ID Number 值和 **PRODUCTS** 表中 Vendor ID Number 值的大小。
4. 允许的比较元素还设置为“Value Expression (值表达式)—Equals”, 这样用户就能比较 Vendor ID Number 值和任意数值的大小。

图 9.9 所示为通用 State 字段的完整通用字段说明。

The screenshot shows the 'Logical Elements' dialog box for the 'State' field. The 'Comparisons Allowed' section is expanded, showing the following options:

Comparison Type	Field	Operator	Value
Same Field	State	=	1
	State	>	2
	State	<	3
Other Fields	State	=	4
	State	>	5
	State	<	6
Value Expression	State	=	7
	State	>	8
	State	<	9

图 9.9 通用 State 字段的完整通用字段说明

FIELD SPECIFICATIONS							
General Elements							
Field Name:	State	Specification Type:	<input type="checkbox"/> Unique <input checked="" type="checkbox"/> Generic <input type="checkbox"/> Replica				
Parent Table:		Source Specification:					
Label:							
Shared By:							
Alias(es):							
Description:	A state or territory within the United States in which a person, organization, or institution resides or conducts business.						
Physical Elements							
Data Type:	Alphanumeric	Character Support:					
Length:	2	<input checked="" type="checkbox"/> Letters (A-Z) <input type="checkbox"/> Keyboard (., / \$ # %)					
Decimal Places:	None	<input type="checkbox"/> Numbers (0-9) <input type="checkbox"/> Special (© ® ™ Σ π)					
Input Mask:	AA						
Display Format:	Both letters should be capitalized.						
Logical Elements							
Key Type:	<input checked="" type="checkbox"/> Non <input type="checkbox"/> Primary <input type="checkbox"/> Foreign <input type="checkbox"/> Alternate	Edit Rule: <input checked="" type="checkbox"/> Enter Now, Edits Allowed <input type="checkbox"/> Enter Now, Edits Not Allowed <input type="checkbox"/> Enter Later, Edits Allowed <input type="checkbox"/> Enter Later, Edits Not Allowed <input type="checkbox"/> Not Determined At This Time					
Key Structure:	<input type="checkbox"/> Simple <input type="checkbox"/> Composite						
Uniqueness:	<input checked="" type="checkbox"/> Non-unique <input type="checkbox"/> Unique						
Null Support:	<input type="checkbox"/> Nulls Allowed <input checked="" type="checkbox"/> No Nulls						
Values Entered By:	<input checked="" type="checkbox"/> User <input type="checkbox"/> System						
Required Value:	<input type="checkbox"/> No <input checked="" type="checkbox"/> Yes						
Default Value:							
Range of Values:	All state abbreviations recognized by the United States Postal Service.						
Comparisons Allowed:							
<input type="checkbox"/> Same Field	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/> ≠	<input type="checkbox"/> <	<input type="checkbox"/> <=
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/> ≠	<input type="checkbox"/> <	<input type="checkbox"/> <=
<input checked="" type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input checked="" type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/> ≠	<input type="checkbox"/> <	<input type="checkbox"/> <=
Operations Allowed:							
<input type="checkbox"/> Same Field	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation	
<input checked="" type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input checked="" type="checkbox"/> Concatenation	
<input checked="" type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input checked="" type="checkbox"/> Concatenation	

图 9.9 通用 State 字段通用字段说明

注意下列各项。

1. 描述十分常见，正好适用于这种说明。
2. 显示格式的设置采取指令的形式。这表明设置这一元素具有极大的灵活性。
3. 逻辑元素值的范围恰当。
4. 逻辑元素允许的比较设置为“Value Expression—Equals”，这样用户就能比较 State 值和任意两字符字母数字值的大小。
5. 逻辑元素允许的运算设置为“Other Fields(其他字段)—Concatenation(拼接)”，这样用户就可以拼接给定 State 值和另一个字母数字字段的值。
6. 允许的运算元素还设置“Value Expression—Concatenation”，这样用户就可以将给定 State 值和任意字母数字值串联。

现在，以这个字段（及其说明）作为数据库中所有其他州字段的模板。例如，可以基于通用 State 字段创建一个 VendState 字段。然后根据 State 字段的通用说明，为 VendState 字段定义可复制说明。虽然 VendState 字段的可复制说明从 State 字段的通用说明中获取了初始元素设置，但是可以对该可复制说明的任意元素设置进行修改，让 VendState 字段获得完全个性化处理。图 9.10 所示为 VendState 字段个性化的可复制字段说明。



FIELD SPECIFICATIONS			
General Elements			
Field Name:	VendState	Specification Type:	<input type="checkbox"/> Unique <input type="checkbox"/> Generic <input checked="" type="checkbox"/> Replica
Parent Table:	Vendors	Source Specification:	State
Label:	State		
Shared By:			
Alias(es):			
Description: The state in which the vendor's headquarters are located. This data is a component of the vendor's overall mailing address.			
Physical Elements			
Data Type:	Alphanumeric	Character Support:	
Length:	2	<input checked="" type="checkbox"/> Letters (A-Z) <input type="checkbox"/> Keyboard (., / \$ # %)	
Decimal Places:	None	<input type="checkbox"/> Numbers (0-9) <input type="checkbox"/> Special (© ® ™ Σ π)	
Input Mask:	AA		
Display Format: Both letters should be capitalized.			
Logical Elements			
Key Type:	<input checked="" type="checkbox"/> Non <input type="checkbox"/> Foreign <input type="checkbox"/> Primary <input type="checkbox"/> Alternate	Edit Rule:	
Key Structure:	<input type="checkbox"/> Simple <input type="checkbox"/> Composite	<input checked="" type="checkbox"/> Enter Now, Edits Allowed	
Uniqueness:	<input checked="" type="checkbox"/> Non-unique <input type="checkbox"/> Unique	<input type="checkbox"/> Enter Now, Edits Not Allowed	
Null Support:	<input type="checkbox"/> Nulls Allowed <input checked="" type="checkbox"/> No Nulls	<input type="checkbox"/> Enter Later, Edits Allowed	
Values Entered By:	<input checked="" type="checkbox"/> User <input type="checkbox"/> System	<input type="checkbox"/> Enter Later, Edits Not Allowed	
Required Value:	<input type="checkbox"/> No <input checked="" type="checkbox"/> Yes	<input type="checkbox"/> Not Determined At This Time	
Default Value:	WA		
Range of Values: CA, ID, MT, OR, WA			
Comparisons Allowed:			
<input type="checkbox"/> Same Field	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
<input checked="" type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input checked="" type="checkbox"/> =	<input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
Operations Allowed:			
<input type="checkbox"/> Same Field	<input type="checkbox"/> All	<input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x	<input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation
<input checked="" type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x	<input type="checkbox"/> ÷ <input checked="" type="checkbox"/> Concatenation
<input checked="" type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x	<input type="checkbox"/> ÷ <input checked="" type="checkbox"/> Concatenation

图 9.10 VendState 字段个性化的可复制字段说明

对于这一说明，需要注意以下几点。

1. 字段名称 (VendState) 准确诠释了该字段所表示的含义。
2. 标签 (“State”) 将出现在屏幕和打印文档中。
3. 一般元素源说明适当参考了 State 字段的通用说明。
4. 描述元素专为该字段设置。前面曾提到，源说明中描述更为通用。
5. 默认值专为该字段设置；源说明中无默认值。
6. 值的范围元素也是专为该字段设置；源说明中值的范围更为宽泛。

下一章将介绍如何为外键字段定义可复制字段说明。

定义每个字段的字段说明

现在，所有必要字段都已经分配到每个表，读者也了解了字段说明中的各个元素，可以着手为数据库中每个字段定义字段说明了。虽然完成一个环节需要投入大量的时间，但是记住你所花费的辛劳能确保数据一致、有效并且错误少，是为了建立字段级完整性。付出的汗水将得到高额回报，因为从数据库中检索到的信息将始终及时准确，等到在 RDBMS 程序中实现数据库时，也将拥有一整套可靠的结构蓝图。

通过与用户和管理人员一道定义字段说明，就能确保该说明达到最为完善和准确的水平。他们能提供对数据的深刻见解，对完善该说明的逻辑元素也会有特别的帮助。当然，不必与机构内部所有人一一交谈，不过确实需要将一部分代表性的人员召集起来，与之商谈。这些人就是非常熟悉这些数据及其用途的人群。商谈的次数和时间，应视完成访谈过程的具体要求而定，访谈应尽量全面。尤其是，切勿草率了事。这样做只会损害整体工作的成果，增加犯下过失的风险。

最佳的策略就是尽量定义更多更完整说明，然后与参与者一道完成剩余工作。制订字段说明时，应运用最准确的判断为每个元素定义设置。即使设

置有些微谬误，或者无法为某些元素提供设置，也不必担忧。因为无论情况如何，你都将和参与者一道对其进行评审。在为熟悉的所有字段定义好说明之后，就可以开始与参与者会谈，为剩余的字段制定说明。

在初次会谈期间，首先就要解释字段说明中的各个元素，确保所有人都有全面认识。向参与者简单介绍说明的各个元素，他们就知道如何恰当地帮助定义说明。（在后续的会谈中，直接评审这些元素，确保所有人都记住它们所表示的意义。）

接下来，评审已定义的所有说明，让参与者评价元素的设置是否合适正确。在某些情况下，参与者会揭露出某个字段的新信息，影响到该字段的说明。例如，一位参与者可能记起（受讨论中某个主题的启发），一组特殊的值过去始终被用于特定字段；因此，设置该字段值的范围元素，反应这一新信息。确保检查到说明的每一个部分，待参与者不再提出改进建议，就可以进入下一说明评审工作。其余说明都要重复这一过程。

现在，和参与者一道解决无法定义或完成的字段。尽量选择对这些字段最为熟悉的人参与讨论，因为他们有可能清楚逻辑元素类别应该使用什么设置。为每个字段确定合适的元素设置，并在字段说明表中标明。完成数据库中每一个字段说明，这一过程也就结束了。

新数据库的设计现在已接近尾声。下一章将介绍如何建立数据库中表之间的关系。关系之所以重要，是因为有了关系，就可以视图从多个表中同时提取数据。

案例分析

既然所有适宜的字段都已经分配到迈克单车行数据库的表中，那么就可以为字段定义字段说明了。与迈克和他的员工会谈之前，应该尽可能多地定义字段说明。所有表都是常见的类型，字段也相当明确，所以轻易就能将说明定义好。图 9.11 所示为 PRODUCTS 表中 Product Description 字段的说明。

FIELD SPECIFICATIONS			
General Elements			
Field Name:	Product Description	Specification Type:	<input checked="" type="checkbox"/> Unique <input type="checkbox"/> Generic <input type="checkbox"/> Replica
Parent Table:	Products	Source Specification:	
Label:	Description		
Shared By:			
Alias(es):			
Description:	A statement that provides pertinent details about the product. This information is useful to our sales and promotion efforts and is provided to our customers by means of various promotional materials.		
Physical Elements			
Data Type:	Alphanumeric <input checked="" type="checkbox"/>	Character Support:	<input type="checkbox"/> Foreign
Length:	180	<input checked="" type="checkbox"/> Letters (A-Z)	<input checked="" type="checkbox"/> Keyboard (., / \$ # %)
Decimal Places:	None	<input checked="" type="checkbox"/> Numbers (0-9)	<input checked="" type="checkbox"/> Special (© ® ™ Σ π)
Input Mask:			
Display Format:			
Logical Elements			
Key Type:	<input checked="" type="checkbox"/> Non <input type="checkbox"/> Primary	Edit Rule:	<input checked="" type="checkbox"/> Enter Now, Edits Allowed
	<input type="checkbox"/> Foreign <input type="checkbox"/> Alternate		<input type="checkbox"/> Enter Now, Edits Not Allowed
Key Structure:	<input type="checkbox"/> Simple <input type="checkbox"/> Composite		<input type="checkbox"/> Enter Later, Edits Allowed
Uniqueness:	<input type="checkbox"/> Non-unique <input checked="" type="checkbox"/> Unique		<input type="checkbox"/> Enter Later, Edits Not Allowed
Null Support:	<input type="checkbox"/> Nulls Allowed <input checked="" type="checkbox"/> No Nulls		<input type="checkbox"/> Not Determined At This Time
Values Entered By:	<input checked="" type="checkbox"/> User <input type="checkbox"/> System		
Required Value:	<input type="checkbox"/> No <input checked="" type="checkbox"/> Yes		
Default Value:			
Range of Values:			
Comparisons Allowed:			
<input type="checkbox"/> Same Field	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
<input type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
Operations Allowed:			
<input type="checkbox"/> Same Field	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation
<input type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation

图 9.11 Product Description 字段的字段说明

现在，你和迈克还有他的员工讨论已定义的字段说明。无人对说明提出异议：大家都认定所有元素设置看上去合适正确。不过，你对 **PRODUCTS** 表的 **Category** 字段有疑问：你想知道值的范围元素的合理设置。大家意见不一，没有人清楚该字段完整有效的分类列表。于是，你决定暂时制订一个一般值的范围。

等到为该数据库建立业务规则时，再次回到这一字段（及其元素）。解决好这个问题，会谈以及整个定义字段说明的过程也就结束了。

Logical Elements									
Key Type:	<input checked="" type="checkbox"/> Non	<input type="checkbox"/> Primary	Edit Rule:						
	<input type="checkbox"/> Foreign	<input type="checkbox"/> Alternate	<input checked="" type="checkbox"/> Enter Now, Edits Allowed						
Key Structure:	<input type="checkbox"/> Simple	<input type="checkbox"/> Composite	<input type="checkbox"/> Enter Now, Edits Not Allowed						
Uniqueness:	<input checked="" type="checkbox"/> Non-unique	<input type="checkbox"/> Unique	<input type="checkbox"/> Enter Later, Edits Allowed						
Null Support:	<input type="checkbox"/> Nulls Allowed	<input checked="" type="checkbox"/> No Nulls	<input type="checkbox"/> Enter Later, Edits Not Allowed						
Values Entered By:	<input checked="" type="checkbox"/> User	<input type="checkbox"/> System	<input type="checkbox"/> Not Determined At This Time						
Required Value:	<input type="checkbox"/> No	<input checked="" type="checkbox"/> Yes							
Default Value:									
Range of Values:	Any valid internal or external product category.								
Comparisons Allowed:									
<input type="checkbox"/> Same Field	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/> ≠	<input type="checkbox"/> <	<input type="checkbox"/> <=		
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/> ≠	<input type="checkbox"/> <	<input type="checkbox"/> <=		
<input checked="" type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input checked="" type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/> ≠	<input type="checkbox"/> <	<input type="checkbox"/> <=		
Operations Allowed:									
<input type="checkbox"/> Same Field	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation			
<input checked="" type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input checked="" type="checkbox"/> Concatenation			
<input checked="" type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input checked="" type="checkbox"/> Concatenation			

图 9.12 PRODUCTS 表中 Category 字段的逻辑元素

小结

本章开篇即解释了字段说明重要的原因及其带来的好处。定义说明有助于建立和实施**字段级完整性**，提升整体数据完整性，以及让你全面了解数据库中数据的性质和用途。借助这一层次的认识，能让数据实现最大化利用。

接下来，对字段说明做了剖析。其中介绍了说明中分为三类元素以及用于记录这些元素的表。然后，具体讨论了每一类别的元素。一般元素类别表示字段最基本的属性。在这一类别的讨论中，介绍了一套创建优秀字段说明指南。可以定义三种说明。选用合适的说明，就能创建并维护前后一致的字段定义。接着，讨论了物理元素。物理元素和字段的结构有关。最后讨论了逻辑元素。逻辑元素主要和字段的值有关，这一类别包括的元素有键类型、null 支持、值的范围、编辑规则、允许的比较和允许的运算。

然后讨论的是如何使用每种说明。文中介绍了一套指南，帮助你针对给定字段判断哪一种说明适用。另外，还评审了说明样本，指出了他们的不同之处。

结尾讨论了定义字段说明。确保说明完整准确的最佳方法就是，同用户和管理人员一道定义说明。首先，你应该尽己所能完成一部分，然后再和员工一起定义剩余的字段说明。也可以和员工一同改进自己起初定义的说明。

思考题

1. 说出字段说明重要的两个主要原因。
2. 建立字段级完整性有何收获？
3. 字段说明中的三类元素是什么？
4. 说出三类说明的名称。
5. 创建合适的字段描述有什么好处？
6. 数据类型元素指示什么？
7. 字符支持元素指明什么？
8. 显示格式元素的用途是什么？
9. 字段说明能指示哪种键？



第 10 章

关系数据库系统

表关系

一段完全被视为理所当然的关系，它所提供的慰藉是无可替代的。

——艾瑞斯·梅铎

本章内容

关系为何重要

表的类型

识别现有关系

建立关系

改进所有外键

建立关系特征

关系层次完整性

案例分析

小结

思考题

第3章“术语”曾提到，一表的记录与另一表的记录以某种方式联系起来，这两表之间就存在一种关系。关系有三个特征：表之间关系的类型、每个表参与的方式以及程度。

本章将逐个详细地讨论这些问题。首先，将介绍如何识别和建立数据库中表之间的关系，然后再讨论如何设置每个关系的特征。另外，还将说明如何用图解法表示表和关系，这样能直观展示整个数据库的结构。

关系为何重要

关系是关系数据库的重要组成部分。

- 对于存在逻辑联系的两表，可在两表之间建立连接。两表通过各自表中包含的数据产生逻辑联系。例如，图 10.1 中的两个表。

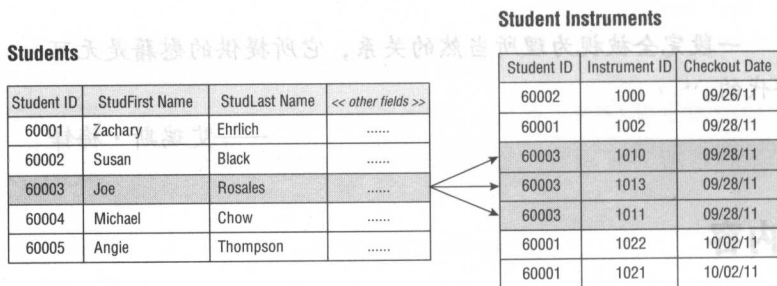


图 10.1 两个具有逻辑联系的表

STUDENTS 表与 STUDENT INSTRUMENTS 表中数据之间存在一种逻辑关系。学生在一学年中可以考核一种或多种乐器，所以 STUDENTS 表中某个记录（代表该学生）可以与 STUDENT INSTRUMENTS（代表该学生考核通过了相应的乐器）。

- 有助于进一步改进表结构和将冗余数据减至最少。建立两表之间的关系时，必定会给两表的结构带来细微的改变。这些改进将使表结构更为合理有效，并将两表所包含的冗余数据降至最低水平。
- 这种机制能实现同时从多个表中提取数据。第 12 章“视图”中将介绍如何在一种关系的前提下，使用多个相关表中的字段建立视图。

定义规范的关系能确保关系层次完整性，这种完整性反过来会保证关系本身完善可靠。（前文曾提到，关系层次完整性是整个数据完整性的重要组

成部分。唯有认真细致地建立好每种关系，才能充分利用关系型数据库的各种优势。草草了事，将难于应付多个表中的数据，而且任何插入、更新和删除相关表中的记录的行为，都必定会产生相应的问题。随着设计过程的展开，你将逐渐了解这些问题。

关系的类型

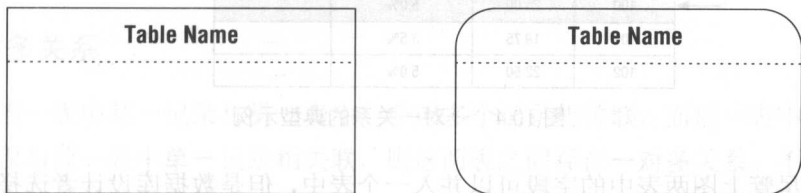
在数据库中建立表之间的关系之前，必须知道表之间可以存在哪些关系。正确识别表关系是成功设计数据库的一项宝贵的技能。

两表之间可存在三种关系：一对一、一对多，以及多对多。任何时候，牵涉其中的表都只具备一种关系。（表之间的关系几乎不发生变化。除非其中一表的结构有重大改变，才有可能改变关系。）

❖注意：讨论每种关系之初都会用到该种关系的一个通用范例。了解如何将关系可视化，通常能让你认识该种关系背后的规律。一旦你理解了该关系运作的方式和原因，就能轻易判断特定表中是否存在这种关系。

讨论中也分别包含了如何将该种关系用图解法展现的示例。本书将酌情提供与图解过程相关的说明，如有必要，也将一并解释图解中包含的符号。这样，就能够有序地学习图解法，不必急于一次记住整套图解符号。

图 10.2 所示为图解表关系首先将用到的符号。



数据表

子集表

图 10.2 数据表和子集表的图解符号

一对一关系

当一表中某一记录仅与另一表中单一记录相关联，且反之亦然，则这两表之间就存在一对一关系。图 10.3 所示为一对一关系的通用范例。

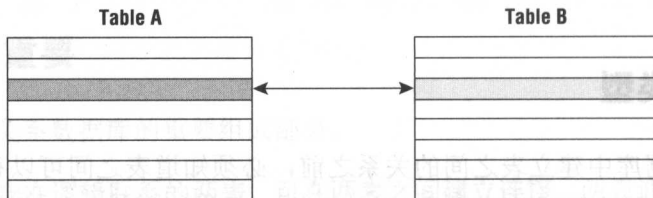


图 10.3 一对一关系的通用范例

如你所见，TABLE A 中一记录仅与 TABLE B 中一记录关联，TABLE B 中也仅有一记录与 TABLE A 中一记录关联。通常，一对一关系牵涉到一个子集表。图 10.4 所示为典型的一对一关系，它一般出现在机构人力部门的数据库中。这个示例也展示了一种无子集表的情况。

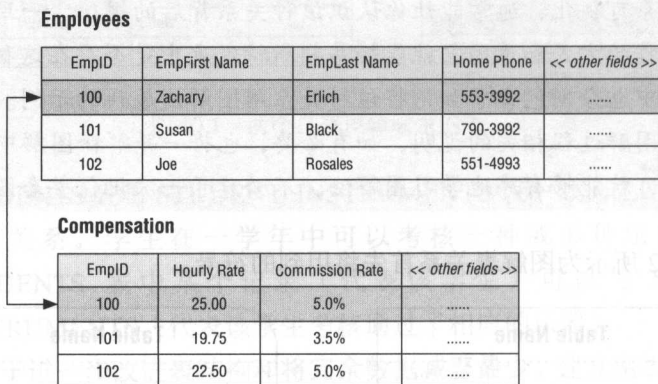


图 10.4 一对一关系的典型示例

尽管上图两表中的字段可以并入一个表中，但是数据库设计者选择将机构内部所有人都可以查看的字段安排在 EMPLOYEES 表中，将只能由授权人员访问的字段放置在 COMPENSATION 表中。只有一个记录需要存储员工的薪酬数据，所以 EMPLOYEES 表中的一记录与 COMPENSATION 表中的另一记录存在明显的一对一关系。

图 10.5 所示为创建一对一关系的关系图解示例。

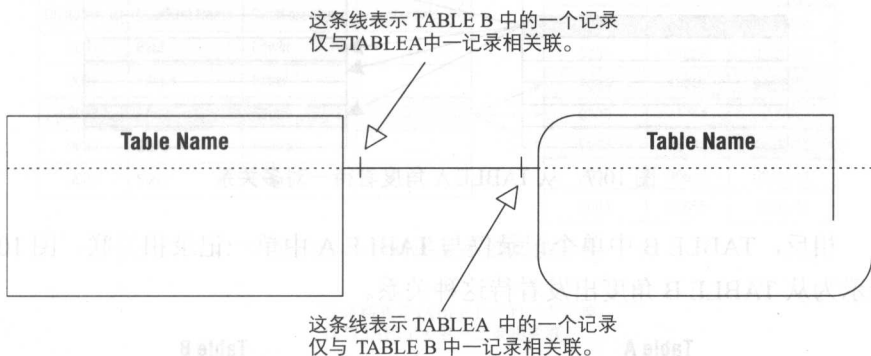


图 10.5 一对一关系示意图

图中两表之间的线条表示关系的类型，每种关系分别使用特定的线条。本章后面的部分将介绍如何利用线条展示关系的特征。图 10.6 所示为图 10.4 中 EMPLOYEES 表和 COMPENSATION 表的关系示意图。（注意，每个表都使用的是数据表符号。）

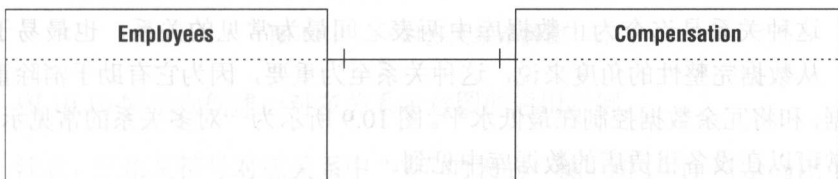


图 10.6 EMPLOYEES 表和 COMPENSATION 表的关系示意图

一对多关系

当一表中某一记录与另一表中一个或多个记录相关联，而后一表中单个记录仅与前一表中单一记录相关联，则这两表之间存在一对多关系。不妨看看这类关系的通用示例吧。

假设，TABLE A 和 TABLE B 之间具有一对多关系。由于存在这种关系，所以 TABLE A 中一记录与 TABLE B 中一个或多个记录相关联。图 10.7 所示为从 TABLE A 角度出发看待这种关系。

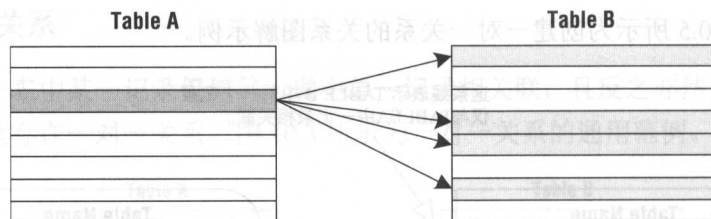


图 10.7 从 TABLE A 角度看待一对多关系

相反, TABLE B 中单个记录仅与 TABLE A 中单一记录相关联。图 10.8 所示为从 TABLE B 角度出发看待这种关系。

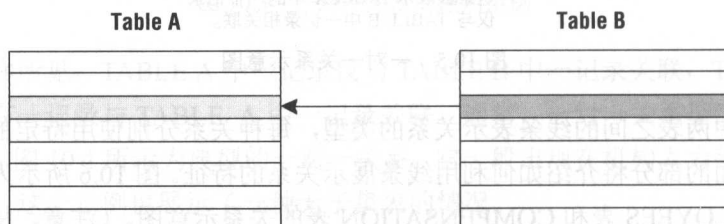


图 10.8 从 TABLE B 角度看待一对多关系

这种关系是迄今为止数据库中两表之间最为常见的关系, 也最易于识别。从数据完整性的角度来说, 这种关系至为重要, 因为它有助于消除重复数据, 并将冗余数据控制在最低水平。图 10.9 所示为一对多关系的常见示例, 经常可以在设备租赁店的数据库中见到。

顾客可以租赁任意的设备, 所以 CUSTOMERS 表中单一记录可以与 CUSTOMER RENTALS 表中一个或多个记录相关联。然而, 在一个时间段单一设备只能与一位顾客关联, 所以 CUSTOMER RENTALS 表中一个记录只能与 CUSTOMERS 表中一个记录相关联。

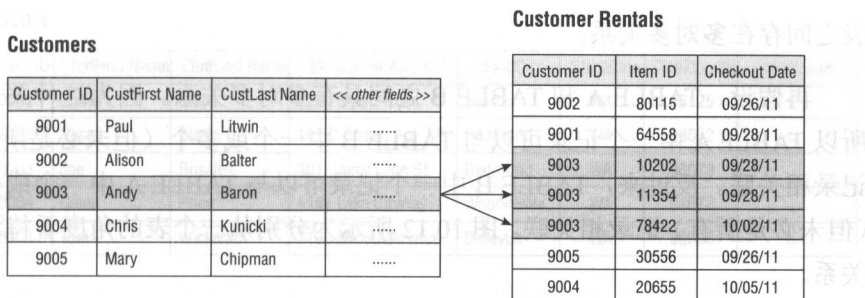


图 10.9 一对多关系的典型示例

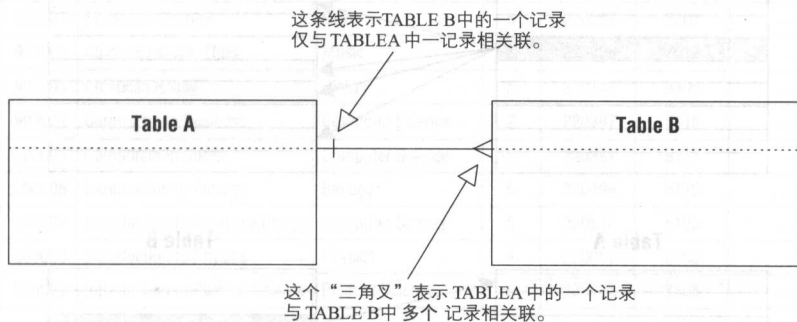


图 10.10 一对多关系示意图

图 10.10 所示为创建一对多关系示意图的通用示例。

注意，三角叉符号对应关系中“多”的表的一侧。图 10.11 所示为图 10.9 中 CUSTOMERS 表和 CUSTOMER RENTALS 表的关系示意图。

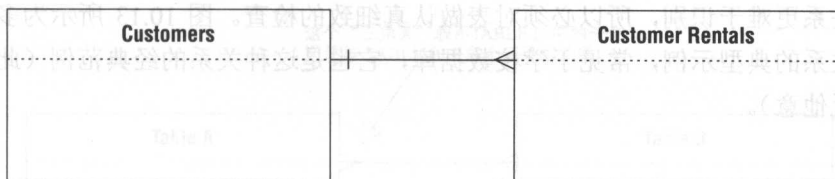


图 10.11 CUSTOMERS 表和 CUSTOMER RENTALS 表的关系示意图

多对多关系

当一表中一记录与另一表中一个或多个记录相关联且反之亦然，则这两

表之间存在多对多关系。

再假设，TABLE A 和 TABLE B 之间具有多对多关系。因为这种关系，所以 TABLE A 中一个记录可以与 TABLE B 中一个或多个（但未必是所有）记录相关联。反过来，TABLE B 中一个记录可以与 TABLE A 中一个或多个（但未必是所有）记录相关联。图 10.12 所示为分别从一个表的角度看待这种关系。

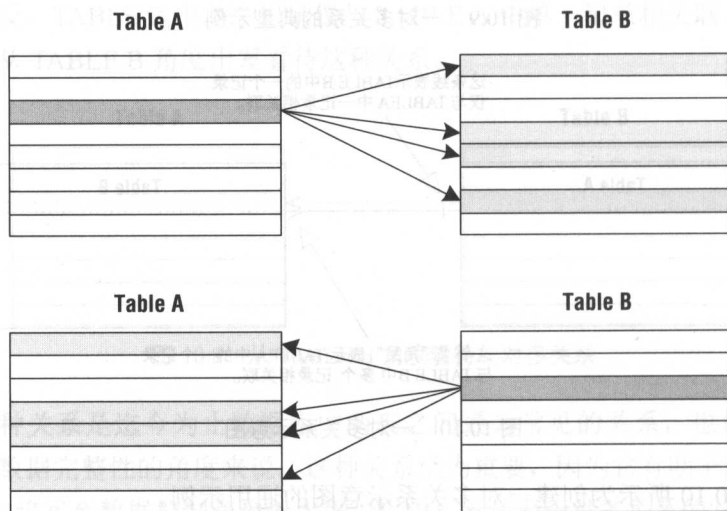


图 10.12 分别从 TABLE A 和 TABLE B 角度看待多对多关系

这是数据库中两表之间存在的第二种常见关系。相比一对多关系，多对多关系更难于识别，所以必须对表做认真细致的检查。图 10.13 所示为多对多关系的典型示例，常见于学校数据库，它也是这种关系的经典范例（此处别无他意）。

Students

Student ID	StudFirst Name	StudLast Name	StudStreet Address	StudCity	StudState	StudZipcode	<< other fields >>
60001	Zachary	Erich	1204 Bryant Road	Seattle	WA	98125
60002	Susan	Black	101 C Street, Apt. 32	Redmond	WA	98052
60003	Joe	Rosales	201 Chery Lane SE	Redmond	WA	98073
60004	Diana	Price	4141 Lake CityWay	Woodinville	WA	98072
60005	Tom	Wickerath	2100 Mineola Avenue	Bellevue	WA	98006

Classes

Class ID	Class Name	Class Category	Credits	Instructor ID	Classroom	<< other fields >>
900001	Advanced Calculus	Math	5	220087	2201
900002	Advanced Music Ther	Music	3	220039	7012
900003	American Histoy	History	5	220148	3305
900004	Computers in Business	Computer Science	2	220387	5115
900005	Computers in Society	Computer Science	2	220387	5117
900006	Introduction to Biology	Biology	5	220498	3112
900007	Introduction to Database Design	Computer Science	5	220516	5105
900008	Introduction to Physics	Physics	4	220087	2205
900009	Introduction to Political Science	Political Science	5	220337	3308

图 10.13 多对多关系的典型示例

一学年中，学生可以选修一门或多门课程，所以 STUDENTS 表中单一记录可以与 CLASSES 表中一个或多个记录相关联。反之亦然，所以 CLASSES 表中单一记录可以与 STUDENTS 表中一个或多个记录相关联。

图 10.14 所示为创建多对多关系示意图的通用示例。

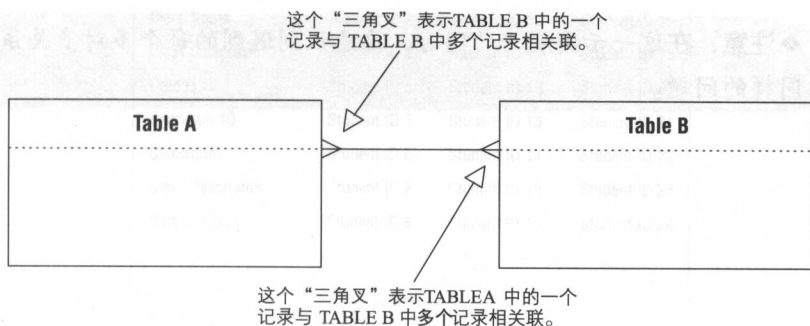


图 10.14 多对多关系示意图

在这种情况下,每个表旁边都分别有一个三角叉。图 10.15 所示为图 10.13 中 STUDENTS 表和 CLASSES 表的关系示意图。

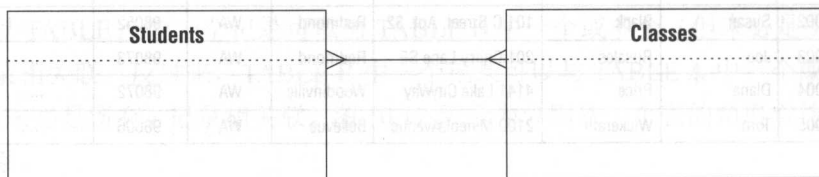


图 10.15 STUDENTS 表和 CLASSES 表的关系示意图

多对多关系存在的问题

多对多关系具有一个固有的难点,必须先解决这个问题,才能使用与这种关系相关的表中的数据。问题就是,如何才能轻松将一表中的记录与另一表中的记录联系起来,从而建立关系。这个问题很重要,因为如果不正确建立关系,就将遇到以下问题。

- 从其中的表检索信息将变得烦琐,甚至困难。
- 一表中将包含大量冗余数据。
- 两表中都存在重复数据。
- 难以插入、更新和删除数据。

经验不足的开发者一般会使用两种方法解决这一状况,但都徒劳无功。本书将以图 10.16 中 STUDENTS 表和 CLASSES 表为例展示使用这两种方法的结果。

❖注意:在这一示例展开的同时,要意识到遇到的每个多对多关系都存在同样的问题。

Table Structures			
Students		Classes	
Student ID	PK	Class ID	PK
StudFirst Name		Class Name	
StudLast Name		Class Category	
StudStreet Address		Credits	
StudCity		Instructor ID	
StudState		Classroom	
StudZipcode		Class Description	
StudHome Phone		Catalog Code	
StudEmail Address			
Social Security Number			

图 10.16 STUDENTS 表和 CLASSES 表的结构

可以看到，两表之间无实际连接，所以无法将一表中的记录与另一表中的记录联系起来。为建立连接可能使用的第一种方法是，从一表中取一个字段，根据相应次数将它逐个添加到另一表。（习惯使用电子表格的人通常使用这种方法。）例如，从 STUDENTS 表中选取 Student ID 字段，将它添加到 CLASSES 表结构中，同时根据参加课程最大学生数量，创建相同数量的该字段副本。图 10.17 所示即是修改版的 CLASSES 表结构。

Table Structures				
Classes				
Class ID	PK	Student ID 1	Student ID 9	Student ID 17
Class Name		Student ID 2	Student ID 10	Student ID 18
Class Category		Student ID 3	Student ID 11	Student ID 19
Credits		Student ID 4	Student ID 12	Student ID 20
Instructor ID		Student ID 5	Student ID 13	Student ID 21
Classroom		Student ID 6	Student ID 14	Student ID 22
Class Description		Student ID 7	Student ID 15	Student ID 23
Catalog Code		Student ID 8	Student ID 16	Student ID 24

图 10.17 将 Student ID 字段添加到 CLASSES 表结构中

这个结构可能出现問題，所以你也許嘗試從 CLASSES 表中選取 Student ID 字段，將它添加到 STUDENTS 表結構中。圖 10.18 所示為修改版的 STUDENTS 表結構。

Table Structures		
Students		
Student ID	PK	Class ID 1
StudFirst Name		Class ID 2
StudLast Name		Class ID 3
StudStreet Address		Class ID 4
StudCity		Class ID 5
StudState		Class ID 6
StudZipcode		Class ID 7
StudHome Phone		Class ID 8
StudEmail Address		
Social Security Number		

圖 10.18 將 Student ID 字段添加到 STUDENTS 表結構中

這些結構是否看上去似曾相識呢？沒錯。通過使用這種方法，你所做的就是將“扁平化”的多值字段引入表結構中。這樣一來，也引入了與之相關的問題。（如有必要，可回顧一下第 7 章“建立表結構”。）儘管知道如何解決多值字段，但是這並非建立關係的正確途徑。

第二種可能使用的方法只是第一種方法的變種。從一表中選取一個或多個字段，將每個字段的一個例子添加到另一表中。例如，為了確認某學生當前選修的課程，可能從 CLASSES 表中選取 Class ID、Class Name 和 Instructor ID 字段，並將其添加到 STUDENTS 表中。相比第一種方法，它似乎有顯著提升。但是，當在修改後的 STUDENTS 表中加載樣本數據，你會發現這樣的修改也帶來了問題。

Students

Student ID	Student First Name	Student Last Name	Class ID	Class Name	Instructor ID	<< other fields >>
60001	Zachary	Erlch	900009	Introduction to Political Science	220087
60001	Zachary	Erlch	900002	Advanced Music Theory	220039
60001	Zachary	Erlch	900003	American History	220148
60001	Zachary	Erlch	900004	Computers in Business	220121
60002	Susan	Black	900009	Introduction to Political Science	220087
60002	Susan	Black	900002	Advanced Music Theory	220039
60002	Susan	Black	900006	Introduction to Biology	220117
60003	Joe	Rosales	900004	Computers in Business	220121
60003	Joe	Rosales	900001	Advanced Calculus	220101
60003	Joe	Rosales	900008	Introduction to Physics	220075
60004	Diana	Price	900007	Introduction to Database Design	220120

图 10.19 向修改后的 STUDENTS 表加载样本数据

图 10.19 清晰地展示了使用这种方法将带来的问题。

- **表中包含无用重复字段。**第 7 章中已详细介绍所有无用重复字段及其引发的问题，所以不宜使用这种方法。另外，Class Name 和 Instructor ID 字段不适合放在 STUDENTS 表中。Class ID 字段足以识别课程，而你所要做的也只是识别一个学生选修的课程。
- **存在大量冗余数据。**即使将 Class Name 和 Instructor ID 字段从 STUDENTS 表中移除，Class ID 字段仍然会产生大量冗余数据。
- **难以插入新记录。**如果为一门新课程向 STUDENTS 表（而不是 CLASSES 表）中插入一条记录，同时不输入学生数据，那么与该学生相关的字段将为 null，包括 STUDENTS 表的主键（Student ID）。这就违反了主键的要素，因为主键不能为 null；因此，必须先提供合适的主键值，才能向表中插入该记录。
- **难以删除记录。**如果想删除特定学生的记录，而新课程仅有的数据恰好保存在该学生记录中，事情就会变得非常困难。例如，看看戴安娜·普赖斯（Diana Price）的记录。如果戴安娜决定这一学年不选修课程，删除她的记录，就会丢失“数据库设计入门”的数据。这也许不会造成太大的问题。但是如果有人同时忘记将这门课程的数据输入 CLASSES 表中，那情况就大不一样了。一旦删除戴安娜的数据，你

就必须向 CLASSES 表中重新输入该课程的所有数据。

幸运的是，不必担心这些问题，因为将学习建立多对多关系的正确方法。

自联结关系

这种特殊类型的关系并不存在于两表之间，所以本节开始并未提到。它是存在于同一表中记录之间的一种关系。讽刺的是，在整个设计过程中仍会将它当作一种表关系。

当一个表中给定记录与该表中其他记录相关联时，这个表就具有一种自联结关系（也称为递归关系）。类似于两表之间的关系，自联结关系也可以分为一对一，一对多和多对多。

一对一

当一个表中给定记录仅与该表中另一记录相关联时，就存在一对一自联结关系。图 10.20 中 MEMBERS 表就具有这样的关系。在这种情形中，给定会员只能赞助该组织中另一名会员；Sponsor ID 字段存储了作为赞助人的会员的编号。注意，苏珊·布莱克（Susan Black）是汤姆·维克拉斯（Tom Wickerath）的赞助人。

Members

Member ID	MbrFirst Name	MbrLast Name	Sponsor ID	<< other fields >>
1001	Zachary	Erlch	
1002	Susan	Black	1001
1003	Joe	Rosales	
1004	Diana	Price	1003
1005	Tom	Wickerath	1002

图 10.20 一对一自联结关系的示例

图 10.21 所示为这种关系的示意图。

表一侧的线条显示这种关系具有自我参照（或“递归”）性质，也表明关系的类型。

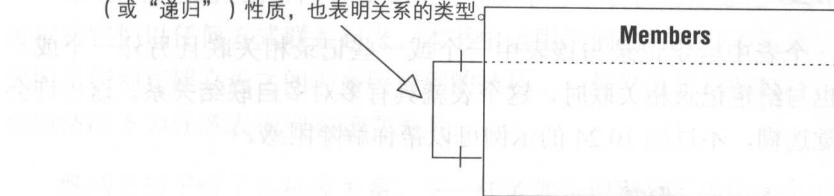


图 10.21 一对一自联结关系示意图

一对多

当一个表中给定记录与该表中另外一个或一些记录相关联时，这个表就具有一对多自联结关系。图 10.22 所示为一个示例，其中给定顾客可以把其他顾客推荐给该公司。Referred By 字段存储的是介绍人的顾客编号。注意，保罗·利特温（Paul Litwin）推荐了安迪·巴伦（Andy Baren）和玛丽·奇普曼（Mary Chipman）。

Customers

Customer ID	CustFirst Name	CustLast Name	Referred By	<< other fields >>
9001	Paul	Litwin	
9002	Alison	Balter	
9003	Andy	Baron	9001
9004	Chris	Kunicki	9003
9005	Mary	Chipman	9001

图 10.22 一对多自我参照关系的示例

图 10.23 所示为一对多自我参照关系的示意图。

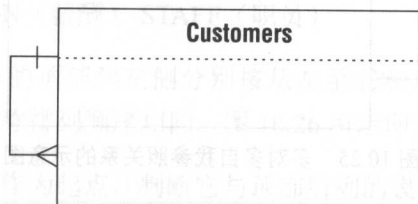


图 10.23 一对多自我参照关系的示意图

多对多

当一个表中给定记录与该表中一个或一些记录相关联且另外一个或一些记录也与给定记录相关联时，这个表就具有多对多自联结关系。这也许会让人感觉迷糊，不过图 10.24 的示例可以帮你解除困惑。

Parts

Part ID	Part Name	<< other fields >>
701	Top Clamp
702	Bottom Clamp
703	Fastening Bolt
704	Clamp Assembly
705	Saddle
706	Seatpost
707	Seat Assembly
708	Body Tube
709	Front Fork Tube
710	Rear Stay Tube
711	Frame Assembly

图 10.24 多对多自我参照关系的示例

在这种情形中，特定部分可以包含多个不同的小部分，其本身也可能是其他部分的一个组成部分。例如，夹具总成（Part ID 704）由紧固螺栓（Part ID 703）、底夹（Part ID 702）和顶夹（Part ID 701）共同组成。另外，夹具总成本身也是座椅总成（Part ID 707）和框架总成（Part ID 711）的组成部分。图 10.25 所示为这种关系的示意图。

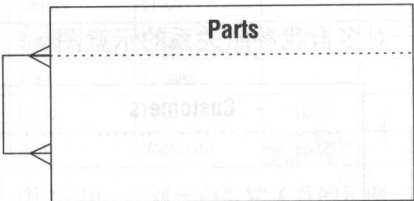


图 10.25 多对多自我参照关系的示意图

❖注意：在继续学习本章剩余部分的示例之前，有必要回顾一下前言中提出的一条原则：

重点在于概念或技巧及其预期结果，而不是用来举证的例子。

无疑,根据这些示例(以及案例分析)中每个表在给定数据库中的角色,可以将它们以任何方式联系起来。本书中运用示例的方式并不重要;真正重要的是识别和建立表之间关系所用到的技巧。一旦学会了这些技巧,就能在任何情况下为任意表识别和建立关系。

既然已经了解了各种表关系,下一任务就是识别数据库中表之间存在的关系。

识别现有关系

在之前编写表描述时(确切地说是第7章),你将用户和管理人员的代表性人群召集起来,帮助完成该任务。这些人也被指定为该机构的代表,将为余下整个数据库设计的决策过程提供帮助。(至少,出于讨论和示例的考虑,当前就是这样设想的。)现在,将再次安排与这些人会谈,让他们帮助识别现有表关系。他们会提供宝贵的意见,因为他们可能对各种主题(或表)的关联有较好的认识。虽然对这些主题关联的方式所做的了解也许并不十分完善和准确,但是他们的贡献仍然对于识别大多数关系能发挥作用。

识别关系之前,先创建一个数据库中所有表的矩阵。(可以在纸、白板或电子表格程序上完成这个步骤。)例如,假设处理的是下列各表:

BUILDINGS (教学楼)	FACULTY (教员)	STUDENTS (学生)
CLASSES (课程)	ROOMS (教室)	
COMPENSATION (薪酬)	STAFF (职员)	

将这些表在矩阵的顶部和左侧分别按从左至右和从上到下的顺序逐个列举出来。确保表名称排列顺序相同。图10.26所示即为该矩阵。

选取左侧一个表作为起点,判断它与顶部所列的表是否有关系,并按照这种方式继续下去。(无论是选择左侧还是顶部的表作为起点都无关紧要。关键是要确保前后使用方式一致,这样就更为简便。)

	Buildings	Classes	Compensation	Faculty	Rooms	Staff	Students
Buildings							
Classes							
Compensation							
Faculty							
Rooms							
Staff							
Students							

图 10.26 建立表矩阵有助于识别现有关系

记住，你寻找的只是**直接**关系，即参与关系的表之间必须具有明确的连接。例如，CLASSES 表与 STUDENTS 表有直接关系，因为特定课程可以由一个或多个学生选修。相反，CLASSES 表与 STAFF 表通过 FACULTY 表建立了**间接**关系；授课的是教员，而不是职员。（当前，无须考虑间接关系。）

处理两个表时，可以向参与者了解表中记录的相关情况。目标是判断一表中的单一记录与另一表中的一个或多个记录的关系，以及反过来两表中记录的关系。（记住，每个记录表示该表所表示主题的一个实例。）当遇到顶部和左侧的表相同时，则判断该表中特定记录与另外一个或一些记录的关系。

可以向参与者提出下列两类问题。

1. **关联型**：这类问题简单直接，通常可以这样提出：某表中的单一记录与另一表中一个或多个记录是否相关联呢？考虑到图 10.26 中的矩阵，可能会问如下问题：

CLASSES 中单一记录与 BUILDINGS 中一个还是多个记录相关联呢？

对以上问题稍作改动，就可以借用来判断某表是否具有自我参照关系：某记录与该表中一个或多个记录是否相关联呢？例如，对 STAFF 表也许会提出这样的问题：

一个职员与其他某个还是某些职员有关联呢？

2. 情境型：这类问题将一表所表示主题的单一实例与另一表所表示主题的多个实例进行对比。它分为两小类：面向归属和面向动作。

a. 面向归属类问题中包含了拥有、属于和包含之类的词语。示例如下：

一张订单中能包含一个还是多个产品呢？

同样，对上述问题稍作改动，也能帮助检验自联结关系。对 PARTS 表可能会提出这样的问题：

一个部件能包含一个还是多个小部件呢？

b. 面向动作类问题中包含了制作、访问、放置、教授和参加之类的行为动词。示例如下：

一位飞行教官能教授一门还是多门课程呢？

如你所料，对上述问题稍加改动，就能用来测试自联结关系：

一位职员能管理一位职员还是多位职员呢？

运用以上问题中你认为最适宜的，识别当前处理的表之间关系。当逐步识别矩阵中表关系，最终会发现这个过程重复了一次：一次是从一表的角度，一次是从另一表的角度。这两次的答案都能识别这两表之间存在的关系。

继续讨论开始的例子，假设决定从 CLASSES 表入手，第一个问题如下：

一门课程在一个还是多个教学楼中开设呢？

答案将从 CLASSES 表角度揭示这两表之间存在的关系。如果得到的是如下回答，则两表之间存在一对一关系。

一门课程只在一个教学楼中开设。

不过，如果得到的是下列回答，则两表之间存在一对多关系。

一门课程可能在多个教学楼中开设。

一旦关系得到确定，就在 CLASSES 表横行（左侧）和 BUILDINGS 表

纵列（顶部）交会的方格中表明关系的类型。可以使用下列速记符号表示相应关系类型：

1:1——一对一

1:N——一对多

M:N——多对多

❖注意：当前，无须使用多对多速记符号，在此列出仅为完整起见。

图 10.27 所示为识别 CLASSES 表关系后的该表矩阵。记住，其中标明的关系是从 CLASSES 表角度来看的。

	Buildings	Classes	Compensation	Faculty	Rooms	Staff	Students
Buildings							
Classes	1:1			1:N	1:1		1:N
Compensation							
Faculty							
Rooms							
Staff							
Students							

图 10.27 完整的 CLASSES 表的表矩阵条目

现在，对矩阵左侧的剩余表逐个重复以上过程。记住，可以从任意表开始。不妨继续来看 BUILDINGS 表，先识别它与 CLASSES 表之间的关系。不错，你早已对此识别过一次，不过是从 BUILDINGS 表的角度看待。假设你提出的问题如下：

一个教学楼可以开设多门课程吗？

如果答案是可以，那么这两表之间就存在一对多关系；否则，就是一对一关系。一旦确认了关系，就要在 BUILDINGS 表横行（左侧）和 CLASSES 表纵列（顶部）的交会方格中表明关系类型。图 10.28 所示为标注 BUILDINGS 表条目后的表矩阵。

	Buildings	Classes	Compensation	Faculty	Rooms	Staff	Students
Buildings							
Classes	1:1			1:N	1:1		1:N
Compensation							
Faculty							
Rooms							
Staff							
Students							

图 10.28 完整的 BUILDINGS 表的表矩阵条目

刚刚已经列举了两个如何识别不同表之间关系的示例。那么，现在不妨来看看如何识别一个表的自联结关系。假设正在处理 STAFF 表，接下来是左侧 STAFF 表和顶部 STAFF 表交会的方格。运用本节之前提到的技巧，也许会提出如下问题：

一个职员能和一个还是多个其他职员相关联呢？

同样，答案将揭示关系的类型。假设你得到的回答是：

特定职员可能是另一职员的配偶。

这表明（相当明显）STAFF 表存在一对一自联结关系。但是，如果得到的回答如下：

特定职员可以管理多位职员。

可能立即就能判断，STAFF 表存在一对多自联结关系。识别这两种关系相对比较容易；而识别多对多自联结关系可能较为困难。

为了判断一个表是否具有多对多自联结关系，必须提出以下问题：一个记录能与一个或一些其他记录相关联，且其中一个还与另外的记录相关联吗？例如，针对 STAFF 表，可能提出如下问题：

一个职员能与一个或多个职员相关联，且这些职员中的一个还能与另外一个或多个职员相关联吗？

类似下列的回答表明 STAFF 表具有多对多自联结关系：

可以,特定职员可以管理多个职员,其中一个职员还可以管理一个或多个其他职员。

一旦该表存在的自联结关系的类型得到确定,就要在表矩阵中标明其关系。

从不同的角度出发,关系往往会发生变化。因此,必须了解如何判断矩阵中每对表之间存在的正式关系。借助下列这套公式可以做出判断;每个公式相当于对应关系类型的定义。(旁边的定义作为参考。)

$1:1 + 1:1 = 1:1$ 当一表中某一记录仅与另一表中单一记录相关联,且反之亦然,则这两表之间就存在一对一关系。

$1:N + 1:1 = 1:N$ 当一表中某一记录与另一表中一个或多个记录相关联,而后一表中单个记录仅与前一表中单一记录相关联,则这两表之间存在一对多关系。

$1:N + 1:N = M:N$ 当一表中一记录与另一表中一个或多个记录相关联且反之亦然,则这两表之间存在多对多关系。

下面列举出识别矩阵中两表之间正式关系所运用的具体步骤。(其中融入了上述关系公式。)不妨先来看看通用版的步骤。

1. 选取两个表,注意两表交会的条目。
2. 在矩阵查找与第一个表同一侧的第二个表,注意它和矩阵另一侧的第一个表交会的条目。
3. 对这两个条目运用相应的公式,识别这两表之间正式关系。
4. 画出该关系的示意图。
5. 在矩阵中划掉这两个条目。

现在,不妨来看看如何将这些步骤运用到矩阵中的两个表上。(在本示例中,从矩阵的左侧入手。)

1. 假设选择的是 BUILDINGS 和 CLASSES 表。注意到这两表交会的条目是 1:N。
2. 在左侧寻找 CLASSES 表，然后注意到 BUILDINGS 和 CLASSES 表交汇的条目是 1:1。
3. 对这些条目运用相应的公式，得出 BUILDINGS 和 CLASSES 表之间的正式关系是 1:N。 $(1:N + 1:1 = 1:N)$
4. 为 BUILDINGS 和 CLASSES 表创建一对多关系示意图。
5. 在矩阵中划掉这两个条目。

图 10.29 所示为其结果。

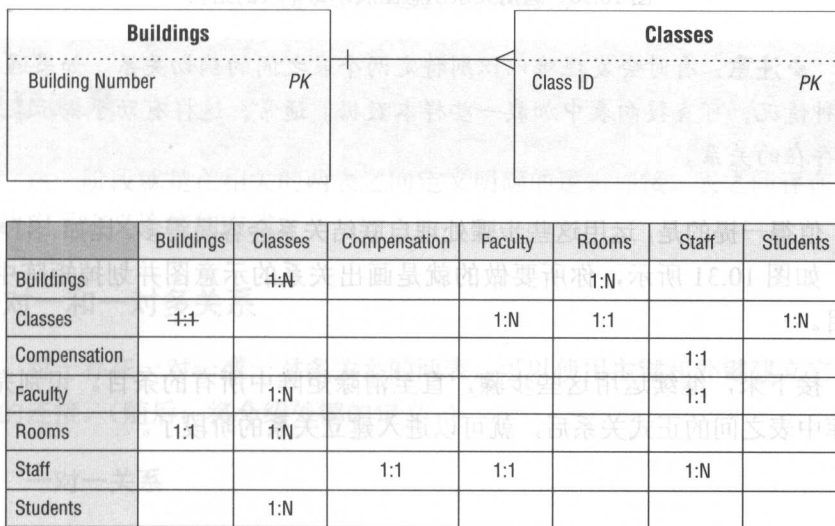


图 10.29 识别 BUILDINGS 和 CLASSES 表之间的正式关系

❖注意：关系示意图是从 BUILDINGS 表角度出发建立的。这是由于 BUILDINGS 表是关系中表示“一”的一端。创建类似的简易示意图时，本书建议始终将“一”的一端安排于关系的左侧，“多”的一端放在右侧。遵循这一做法创建的示意图易于读懂，有助于确保示意图创建方式一致。（然而，创建多个表之间关系的示意图时，这种做法没有作用。）

至少，应该在示意图中包含每个表的主键。这样一来，当你着手建立关系的时候，就能起到直观展示的效果。到目前为止，展示每个表的完整结构就只能到这个地步（如图 10.30 所示）。以这种方式展示结构，往往对强化表之间关系方面的决策有帮助。（本书余下部分将同时使用这两种示意图。）

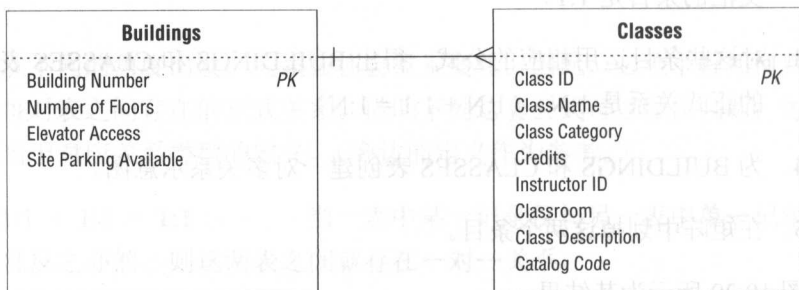
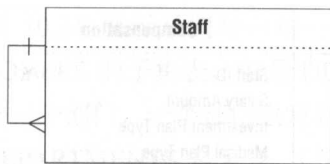


图 10.30 运用关系示意图展示每个表的结构

❖ 注意：有时会发现难以识别特定两个表之间的确切关系。如果遇到这种情况，可直接向表中加载一些样本数据。通常，这样有助于揭示表之间存在的关系。

值得一提的是，运用这些步骤处理自联结关系会容易得多。比如，STAFF 表。如图 10.31 所示，你所要做的就是画出关系的示意图并划掉矩阵中的条目。

接下来，继续运用这些步骤，直至消除矩阵中所有的条目。识别完数据库中表之间的正式关系后，就可以进入建立关系的阶段了。



	Buildings	Classes	Compensation	Faculty	Rooms	Staff	Students
Buildings		1:N			1:N		
Classes	1:1			1:N	1:1		1:N
Compensation						1:1	
Faculty		1:N				1:1	
Rooms	1:1	1:N					
Staff			1:1	1:1		1:N	
Students		1:N					

图 10.31 处理自我参照关系

建立关系

这一阶段就是在相关的两表之间定义明确的逻辑连接。表之间存在的关系决定了定义连接的方式。

一对一和一对多关系

对于存在一对一或一对多关系的两表，可以使用主键和外键建立它们之间的连接。（随后，将介绍外键的定义。）

一对一关系

在这种关系中，一个表作为父表，另一个表作为子表。在向子表输入一条记录之前，父表中必须存在相关记录；换言之，对于子表中的每条记录，父表中必须存在一条相关记录。表所指定的角色通常取决于它所表示的主题，尽管某些情况下角色的指定相当随意。例如，在图 10.32 中，你极有可能将 STAFF 表指定为父表，COMPENSATION 表指定为子表。这种设想合情合理，因为 COMPENSATION 表中如有记录在 STAFF 表中无相关记录，就完全不合逻辑了。

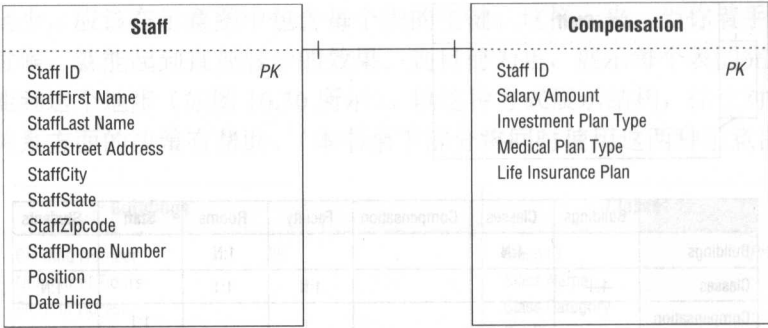


图 10.32 你会选择哪个表作为父表呢？

对于其中一表为子集表的情况，通常会将该子集表指定为子表。不过，也有将子集表指定为父表的情况。

复制父表的主键，将它融入子表的结构中并作为其外键，就建立了一对一的关系。（之所以将之称为外键，是由于子表已有其自身的主键，而从父表中引入的主键对于子表而言是“外来者”。）不过，在大多数一对一关系中，外键也作为子表的主键。

图 10.33 所示为创建 STAFF 和 FACULTY 表之间关系。在这个示例中，STAFF 是父表，因为 FACULTY 表中任一记录必须与 STAFF 表中另一记录相关联；教员本来就属于学校职工。如果你要遵循之前的步骤，就会复制 STAFF 表的主键，将之作为外键添加到 FACULTY 表。然而，这是不必要的，因为 FACULTY 已经是一个定义规范的子集表了。（回顾一下，子集表及其所属的数据表必须共享相同主键。第 7 章和第 8 章“键”已经分别介绍了如何定义子集表和如何建立其主键。）

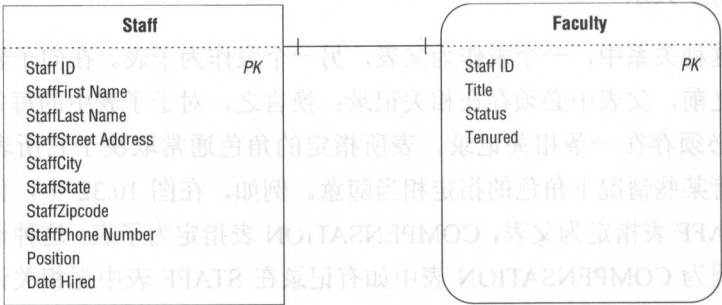


图 10.33 在 STAFF 和 FACULTY 两表之间建立一对一关系

图 10.34 所示示例与一对一关系有细微差别。假设, MANAGERS 是 EMPLOYEES 的子集表, 但是 DEPARTMENTS 与其有直接关系。一个经理只负责一个部门, 一个部门也只有一个经理。进一步假设 MANAGERS 是父表, DEPARTMENTS 是子表。(这个例子中角色的选择相当随意, 且在关系中子集表充当父表。)

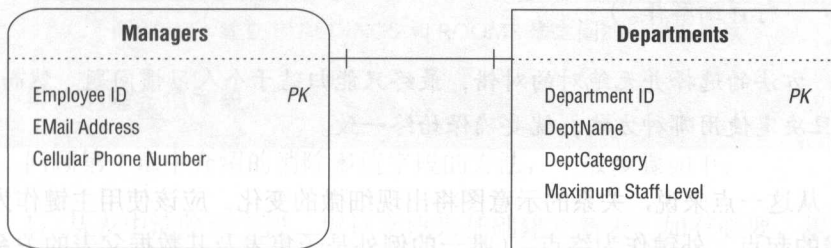


图 10.34 特殊的一对一关系

运用先前的步骤建立这些表之间的关系, 然后识别 DEPARTMENTS 表的新外键 (Employee ID), 在其名称旁标注“FK”。图 10.35 所示为修改后的关系示意图。

一般来说, 只要将这个过程可视化, 就能够建立一对一关系。

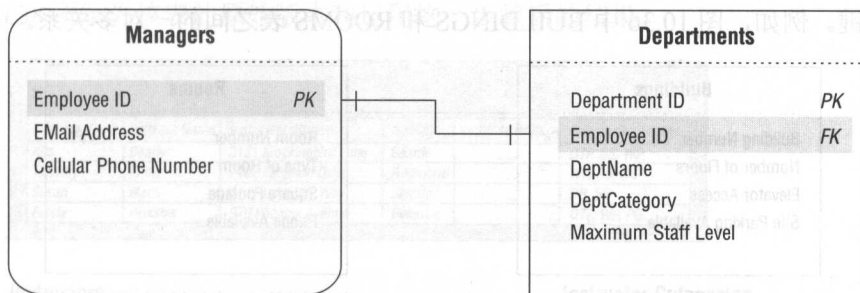


图 10.35 建立 MANAGERS 和 DEPARTMENTS 表之间的关系

❖注意: 许多数据库设计者都使用 ManagerID 作为 MANAGERS 表的主键名称以及 DEPARTMENTS 表的外键名称。本书以 EmployeeID 进行代替有以下原因。

- MANAGERS 是 EMPLOYEES 表的子集表, 所以它与之共享相

同的主键 (Employee ID)。

- 确保该字段符合理想字段的要素。(当它出现在多个表中时, 必须始终保持主要特征不变。)
- 确保该字段符合外键的要素。(本章后面部分将详细介绍外键。)
- 消除对外键本质任何潜在的疑惑。(本书将在外键要素的讨论中进行详细解释。)

方法的选择并无绝对的对错, 最终只能归结于个人习惯问题。然而, 一旦决定使用哪种方法, 就要确保始终一致。

从这一点来说, 关系的示意图将出现细微的变化。应该使用主键作为关系线的起点, 外键作为终点。(唯一的例外是子集表及其数据父表的关系示意图。) 这些细微的改动将有助于更为清晰直观地展现关系, 也更易于识别建立关系的字段。

一对多关系

建立一对多关系所运用的技巧与建立一对一关系关系的技巧相似。直接复制位于关系“一”端的表的主键, 将之加入位于“多”端的表中, 作为其外键。例如, 图 10.36 中 BUILDINGS 和 ROOMS 表之间的一对多关系。

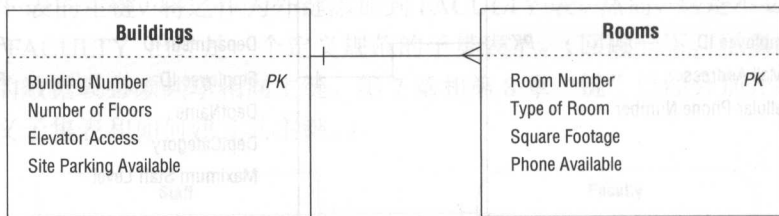


图 10.36 BUILDINGS 和 ROOMS 表之间的一对多关系

这两个表之间的关系是这样的: 一个教学楼可以包含一个或多个教室, 但是一个教室只属于一个教学楼。遵循之前的步骤, 复制 **BUILDINGS** 表的主键 (**Building Number**), 并将之添加到 **ROOMS** 表中, 作为其外键, 就建立了关系。然后, 对其做出与一对一关系相同的调整。图 10.37 所示即为修改后的示意图。(注意, 三角叉符号的中线是重要的连接点, 应直接指向外键。)

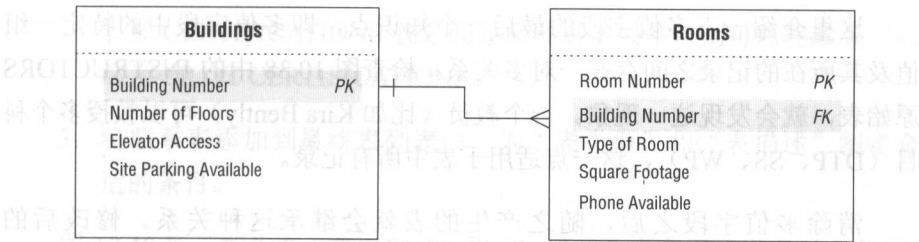


图 10.37 建立 BUILDINGS 和 ROOMS 表之间的一对多关系

再论消除多值字段

回顾第 7 章中介绍的消除多值字段的方法，一般步骤如下：

1. 从表中清除该字段，以该字段为基础建立新表。如有必要，根据本章前面介绍的字段命名指南对该字段进行重新命名。
2. 从原表中采用一个（或一组）字段建立起原表与新表的联系；尽量选取最能表示表主题的字段。所选取的字段将同时存在于两表中。
3. 为新表指定适宜的名称、类型以及描述，并将它添加到最终表列表中。

运用上述步骤消除 INSTRUCTORS 表中多值字段 Categories Taught。图 10.38 所示为该表的原始版本和运用这一步骤后的结果。

Instructors					
InstFirst Name	InstLast Name	InstStreet Address	InstCity	<< other fields >>	Categories Taught
Kira	Bently	3131 Mockingbird Lane	Seattle	DTP, SS, WP
Timothy	Ennis	7402 Kingman Drive	Redmond	WP, DB, OS
Susan	Black	4141 Lake City Way	Seattle	DB, SS
Estela	Rosales	970 Phoenix Avenue	Bellevue	DTP, WP, PG

Instructors

InstFirst Name	InstLast Name	InstStreet Address	InstCity	<< other fields >>
Kira	Bently	3131 Mockingbird Lane	Seattle
Timothy	Ennis	7402 Kingman Drive	Redmond
Susan	Black	4141 Lake City Way	Seattle
Estela	Rosales	970 Phoenix Avenue	Bellevue

Instructor Categories

InstFirst Name	InstLast Name	Category Taught
Kira	Bently	DTP
Kira	Bently	SS
Kira	Bently	WP
Timothy	Ennis	WP
Timothy	Ennis	DB
Timothy	Ennis	OS
Susan	Black	DB
Susan	Black	SS

图 10.38 多值字段 Categories Taught 的原始解决方法

这里介绍一下多值字段的最后一个知识点，即多值字段中的特定一组值及其所在的记录之间存在一对多关系。检查图 10.38 中的 INSTRUCTORS 原始表，就会发现这一现象。一个教员（比如 Kira Bently）可以教授多个科目（DTP、SS、WP），这一点适用于表中所有记录。

消除多值字段之后，随之产生的表就会继承这种关系。修改后的 INSTRUCTORS 和 INSTRUCTOR CATEGORIES 新表显然就是如此。同样地，可以建立这种一对多关系。（当然，前提是为 INSTRUCTORS 表指定了主键。）图 10.39 所示为正确建立这种关系的结果。

Instructors

Instructor ID	InstFirst Name	InstLast Name	InstStreet Address	InstCity	<< other fields >>
60001	Kira	Bently	3131 Mockingbird Lane	Seattle
60002	Timothy	Ennis	7402 Kingman Drive	Redmond
60003	Susan	Black	4141 Lake City Way	Seattle
60004	Estela	Rosales	970 Phoenix Avenue	Bellevue

Instructor Categories

Instructor ID	Category Taught
60001	DTP
60001	SS
60001	WP
60002	WP
60002	DB
60002	OS
60003	DB
60003	SS

图 10.39 INSTRUCTORS 和 INSTRUCTOR CATEGORIES 表之间建立一对多关系

INSTRUCTOR CATEGORIES 表中的 Instructor ID 字段作为外键，有助于建立 INSTRUCTORS 和 INSTRUCTOR CATEGORIES 表之间的一对多关系。Instructor ID 字段也是 INSTRUCTOR CATEGORIES 表复合主键的组成部分；Instructor ID 值和 Category Taught 值的特定组合是识别该表中某个特定记录的唯一方式。

多对多关系

建立多对多关系要借助一个联系表。联系表的创建步骤如下：

1. 复制关系中每个表的主键，使用这些键建立联系表的结构。在联系表中这些字段有两个不同的用途：合并在一起就构成了该表的复合主键，单独分开来每个表都是独立的外键，有助于建立该联系表与其目标之间的关系。
2. 为联系表命名，名称应表明这两表之间关系的性质。例如，在 PILOTS

表和 CERTIFICATIONS 表之间建立多对多关系，则可以将联系表命名为 PILOT CERTIFICATIONS。

3. 将联系表添加到最终表列表中，为“表类型”和“表描述”制定合适的条目。

图 10.40 所示即为在 STUDENTS 和 CLASSES 表之间建立多对多关系。（注意表示联系表所用的示意图符号。）

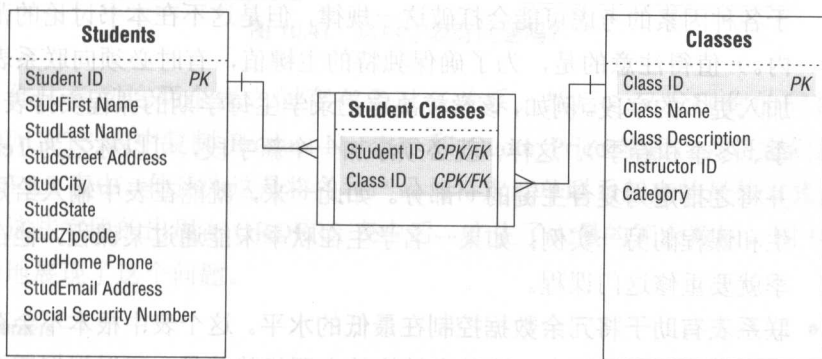


图 10.40 STUDENTS 和 CLASSES 表之间建立多对多关系

❖注意：可以选择 STUDENT SCHEDULES 或 CLASS SCHEDULES 作为该联系表的名称；本书选用 STUDENT CLASSES 仅为作者个人偏好。唯一需要注意的是，应选择对机构最具意义的名称。

创建联系表会产生以下显著结果。

- 原始多对多关系已经被分解，因为 STUDENTS 表和 CLASSES 表之间不再有直接关系。原始关系为两个一对多关系所取代：一个存在于 STUDENTS 和 STUDENT CLASSES 之间，另一个存在于 CLASSES 和 STUDENT CLASSES 之间。在第一个关系中，STUDENTS 表中的单一记录与 STUDENT CLASSES 中一个或多个记录相关联，而 STUDENT CLASSES 中单一记录仅与 STUDENTS 表中的单一记录相关联。在第二个关系中，CLASSES 表中单一记录与 STUDENT CLASSES 中一个或多个记录相关联，而 STUDENT CLASSES 中单一

记录仅与 CLASSES 表中单一记录相关联。

- **STUDENT CLASSES 联系表包含两个外键。**Student ID 和 Class ID 分别是 STUDENTS 和 CLASSES 表复制的主键；因此，根据定义，它们都是外键。同样，它们将有助于建立其父表和联系表之间的关系。
- **STUDENT CLASSES 联系表的复合主键由 STUDENT ID 和 CLASS ID 两字段组成。**除少数情况外，联系表通常包含一个复合主键。（这一规律仅适用于数据库逻辑设计。当逻辑设计转化为物理设计时，出于各种因素的考虑可能会打破这一规律，但是这不在本书讨论的范围内。）值得注意的是，为了确保独特的主键值，有时必须向联系表中加入更多的字段。例如，该学校决定记录学生每学期的课程安排表（秋季、冬季和春季）。这样，就必须添加一个新字段，可以称之为 Term，并将之指定为复合主键的一部分。如此一来，就能在表中输入给定学生和课程的另一实例；如果一名学生在秋季未能通过某课程，他在春季就要重修这门课程。
- **联系表有助于将冗余数据控制在最低的水平。**这个表中根本不会存在多余的数据。实际上，这种表结构的主要优势就是，可以根据需要为单个学生输入尽可能多或者少的课程。为了将它转化为有意义的信息，后续部分将介绍如何同时从这些表中提取数据创建视图。
- **联系表的名称反映出它所帮助建立的关系的用途。**STUDENT CLASSES 表中存储的数据表明了学生及其选修的课程。

处理多对多关系时，为了减少冗余数据和进一步改进关系中涉及的表结构，有时需要向联系表添加字段。假设，你和一位同事一起处理一个新数据库。他让你看看图 10.41 中的 ORDERS 和 PRODUCTS 表。

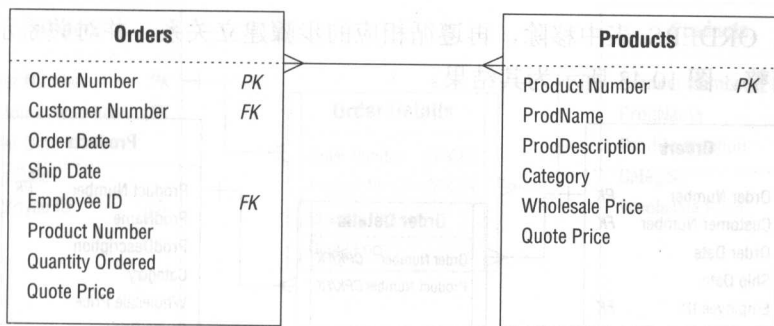


图 10.41 这两个表有问题吗？

你注意到这两个表之间存在多对多关系。然后，又发现你的同事从 PRODUCTS 表中复制 Product Number 和 Quote Price 字段，并将它们加入 ORDERS 表中。他认为这是将各种产品与特定订单联系起来的最佳方式。然而，这些字段的出现在 ORDERS 表中后，产生了大量的冗余数据。图 10.42 清晰地展现了这个问题。

Orders

Order Number	Customer Number	Order Date	<< other fields >>	Product Number	Quantity Ordered	Quote Price
1000	9001	05/16/12	410001	4	8.95
1000	9001	05/16/12	410004	12	3.75
1000	9001	05/16/12	410005	6	5.99
1000	9001	05/16/12	410007	5	6.50
1000	9001	05/16/12	410011	5	6.50
1000	9001	05/16/12	410015	11	4.45
1000	9001	05/16/12	410021	2	31.50
1000	9001	05/16/12	410029	8	5.00
1001	9012	05/16/12	410011	5	6.50
1001	9012	05/16/12	410015	3	4.00
1001	9012	05/16/12	410022	12	6.35

图 10.42 多对多关系建立不当造成的冗余数据

对于给定记录，只能输入一个产品编号、订购数量和报价；因此，必须为顾客订单中的每一项向表中输入一个新记录。例如，顾客编号 9001 在 5 月 16 日所下的订单中包含了 8 个项，所以这一订单在表中就有 8 个记录。

基于对本章之前内容的了解，你知道这种建立关系的方法是不规范的。通过创建和使用联系表，才能正确建立这种关系。所以，将 Product Number

字段从 ORDERS 表中移除，再遵循相应的步骤建立关系，并对关系示意图进行调整。图 10.43 所示为其结果。

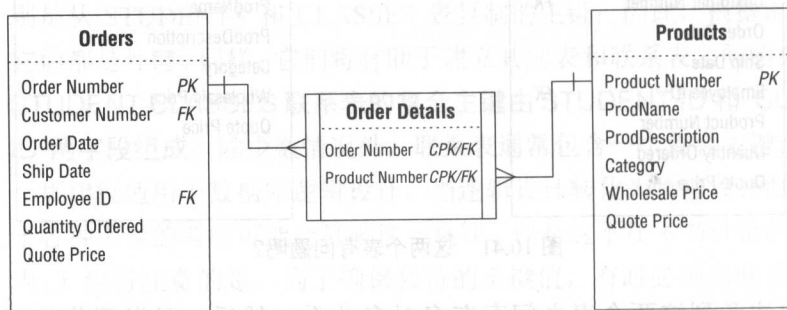


图 10.43 正确建立 ORDERS 和 PRODUCTS 表之间多对多关系

虽然消除了 ORDERS 表中的冗余数据，但是仍然有两个小问题。

1. Quote Price 和 Quantity Ordered 字段不适宜再存在 ORDERS 表中；ORDERS 表的主键不能有效识别它们的值，它们与表中其他字段也没有关系。然而，它们与特定 Product Number 相关，后者是 ORDER DETAILS 表中的给定订单的一部分。
2. 存在重复数据，因为 Quote Price 字段有两个副本：一个在 ORDERS 表中，另一个在 PRODUCTS 表中。

于是，将 ORDER 表中的 Quote Price 和 Quantity Ordered 字段移动到 ORDER DETAILS 表中，就解决了第一个问题。然后，删除 PRODUCTS 表中的 Quote Price 字段；将报价和订单中的产品联系在一起更为合理。最后，修改关系示意图，反映结构的变化。图 10.44 所示即为修改后的示意图。

当建立两表之间的多对多关系时，要确保对每个表进行检查，确认是否有字段应该转移到联系表中。如有疑问，可向所有表中加载样本数据；通常，这样就能揭露出潜在的问题。

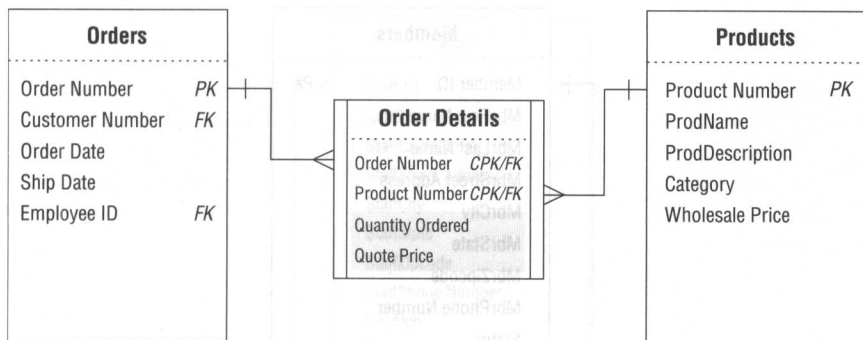


图 10.44 修改后的 ORDER DETAILS 联系表

❖注意：如果忠实地遵循此前学习的设计过程，就不会经常碰到这个问题。不过，直接将现有数据库或遗留数据库中的两个表复制添加到新数据库中，而不对它们的结构进行任何合理的改进，往往就会出现这个问题。另外，和缺乏数据库设计经验的人合作也可能遇到这一问题。

自引用关系

既然已经知道如何建立两表之间的关系，那么建立自引用关系将变得相对容易。

一对一和一对多

建立自引用关系和建立两表之间关系一样，都要使用主键和外键。不过，区别在于这里的外键与主键存在同一表中。往往会发现，外键早已是该表结构的一部分。如果外键并不存在，可以直接创建一个。

不妨重新讨论一下图 10.20 中的 MEMBERS 表。这个表存在一种一对一自引用关系，因为给定某会员只能赞助一位其他会员；Sponsor ID 字段存储的是作为赞助人的会员编号。因为 Sponsor ID 字段的值是专门从 Member ID 字段中提取的，所以它充当着这一关系的外键。将 Sponsor ID 字段指定为正式的外键，并在关系示意图中标注其符号，就建立了关系。图 10.45 所示为修改后的 MEMBERS 表关系示意图。

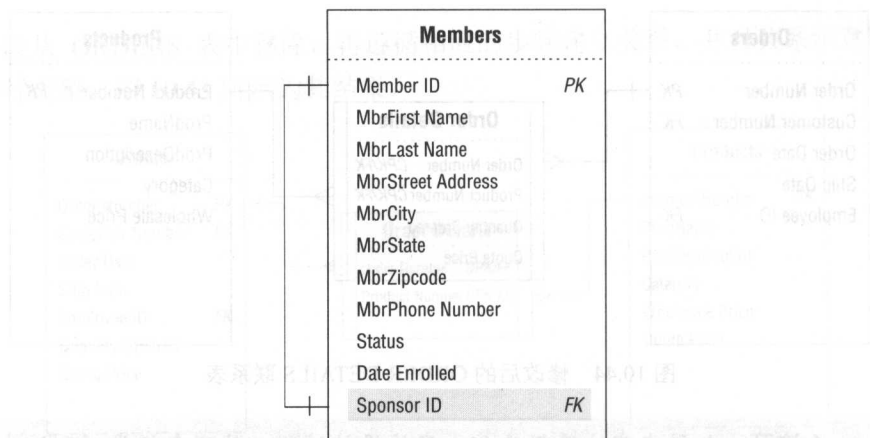


图 10.45 建立 MEMBERS 表一对一自我参考关系

再来考虑一下图 10.46 所示的 STAFF 表示例。你可能会记得，这个表具有一对多自引用关系，因为单一职员能管理一个或多个其他职员。

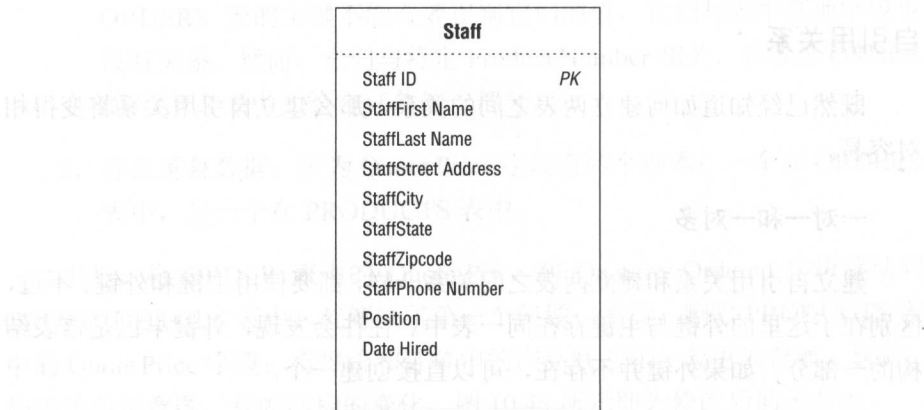


图 10.46 STAFF 表的当前结构

但是，当前给定某职员无法与表中其他职员形成联系；因此，必须创建一个字段，由它充当外键，并建立起关系。不妨假设创建了一个外键字段 **Manager ID**，它的值完全从 **Staff ID** 字段中复制得来。然后，将 **Manager ID** 字段指定为正式的外键，并在关系示意图中标注出来，从而建立好关系。图 10.47 所示为修改后的 STAFF 表关系示意图。

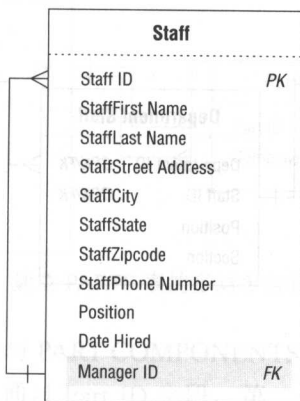


图 10.47 修改后的 STAFF 表

读者可能注意到了，关系线的“一”端指向 MANAGER ID 字段，“多”端指向 Staff ID 字段。这一点是完全可以接受的，因为一位经理管理一个或多个职员，特定职员只会向一位经理汇报。（关系线的“一”端通常指向主键，“多”端指向外键。）

处理一对一和一对多自引用关系时，要花时间认真检查每个表的结构。有时，出于消除关系的考虑，会发现能够（抑或需要）修改和提升现有结构。但是，你可能会问，“为什么要这么做呢？”

从参与这些关系的表中检索信息，过程乏味且有一定难度。（本书不讨论其原因。）另外，关系的存在意味着需要增添新字段和表结构。

再次考虑一下 STAFF 表。是不是觉得：如果需要记录担任经理的职员，也就需要记录他们所管理的部门呢？假如果真如此，那么数据库就必须记录部门的其他方面。应该与相应职员进行短暂的会谈，充分了解相关情况，然后采取相应的措施。

不妨假设你的想法是对的，机构确实希望记录部门数据。图 10.48 所示为你可能采取的一种措施。

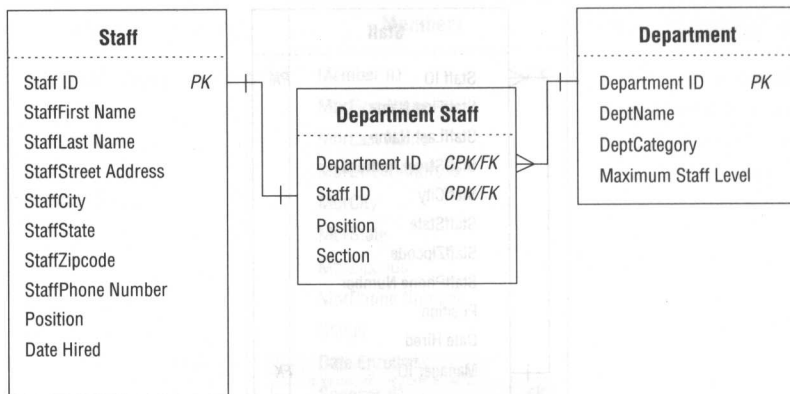


图 10.48 消除自我参考关系并添加新结构记录部门数据的结果

这些新的结构和关系能高效记录数据，并提供与部门有关的各种信息。（当然，必须确保新的字段和表符合之前提到的各种设计要素。）

值得注意的是，设计规范数据库中可能存在自引用关系。不过，必须对此保持谨慎，确保所有自引用关系切实发挥作用。

多对多

与两表之间的多对多关系情况相同，建立这种自引用关系也要使用联系表。稍有不同的是，此处建立联系表所使用的字段取自相同的父表。

不妨来重新考虑一下图 10.24 中的 PARTS 表。这个表具有多对多自引用关系，原因在于特定部件可以由多个不同的小部件构成，而该部件本身也可能是其他部件的组成部分。和建立其他多对多关系一样，这里需要使用一个联系表。鉴于当前给定部件无法与表中其他部件联系起来，所以必须创建一个新字段。比如，创建一个字段 Component ID。这个字段存储的是作为母件组成部分的部件的编号。这样，就可以使用 Part ID 和 Component ID 字段作为基础，建立联系表。出于对示例的考虑，本文将新联系表称为 PART COMPONENTS。一旦创建和命名完联系表，就要确保对 PARTS 表的关系示意图做相应的修改。图 10.49 所示为其结果。

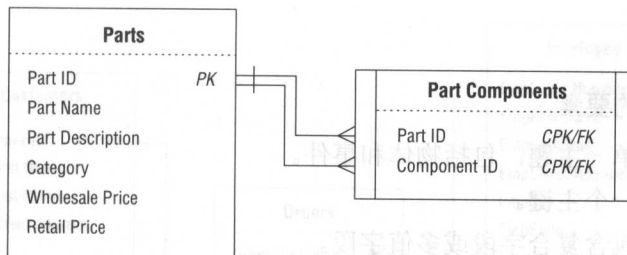


图 10.49 建立 PARTS 表的多对多自我参考关系

可以看到, PARTS 表与 PART COMPONENTS 表有两个不同的一对多关系。第一个关系的建立是通过 Part ID 字段, 第二个是通过 Component ID 字段。图 10.50 展示了这两个关系运作的方式。注意, 夹具总成 (Part ID 704) 包含了三个部件, 它本身也是座椅总成 (Part ID 707) 和框架总成 (Part ID 711) 的组成部分。

现在, 运用刚刚学到的技巧建立数据库中已识别的所有关系。必须确保为每个关系创建一个示意图, 随着设计过程的进一步展开, 将向这些示意图中添加新的信息。

Parts

Part ID	Part Name	<< other fields >>
701	Top Clamp
702	Bottom Clamp
703	Fastening Bolt
704	Clamp Assembly
705	Saddle
706	Seatpost
707	Seat Assembly
708	Body Tube
709	Front Fork Tube
710	Rear Stay Tube
711	Frame Assembly

Part Components

Part ID	Component ID
704	701
704	702
704	703
707	704
707	705
707	706
711	704
711	708
711	709
711	710

图 10.50 PARTS 和 PART COMPONENTS 表之间的数据关系

评审表结构

建立好表之间的关系后, 应评审所有的表结构。记住, 建立关系时, 对现有表结构做了调整并创建了一些新的表结构; 因此, 要确保每个表符合理

想表的要素。

理想表的要素

- 表示单一主题，包括物体和事件。
- 拥有一个主键。
- 不得包含复合字段或多值字段。
- 不得包含计算字段。
- 不得包含无用重复字段。
- 包含的冗余数据量仅为最低水平。

当判断某表不符合理想表的要素时，找出问题并进行必要修改。然后，从该表相应的设计阶段开始审查，直至回到这一阶段。如果严格遵循之前的步骤，表就不应出现任何问题。

改进所有外键

当一个主键用于建立两表之间一对一或一对多关系时，该主键会成为外键。和此前处理的其他所有键一样，外键必须符合一组特定的要素。这些要素一起被称为外键的要素。

外键的要素

- 与相应被复制的主键名称相同。除非有充足理由，否则应尽量遵循这一规则。（回顾第 9 章“字段说明”中别名这一字段元素的讨论。其中提供了打破这一规则的示例。）看看图 10.51 所示关系示意图，注意外键与其参照的主键名称有所不同。

名称不同会造成一个问题，因为无法确定这些外键真正有效且确实是参照这些主键建立的。Emp # 真与 Employee Number 对等吗？“Emp”真是“Employee”的缩写，还是表示其他什么呢？为什么 ORDERS 表中要采用 Client # 而不是 Customer ID 呢？这两者之间有什么区别吗？它们存储的是相同类型的数据吗？在做其他事之前，必须先解答这些问题。

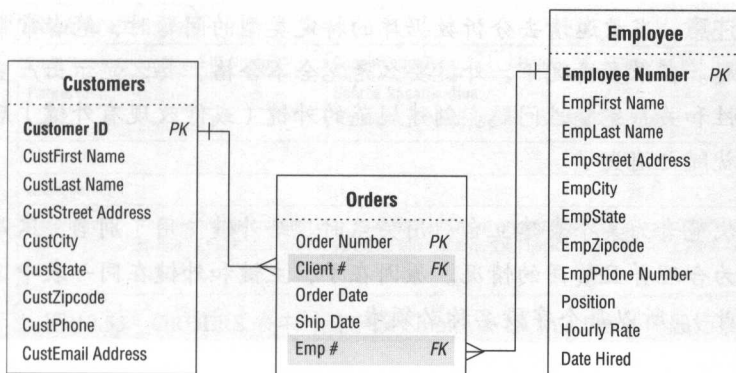


图 10.51 名称不一的主键和外键

也许你会说，它们的名称非常相似，据此可以假设外键有效。如有任何疑虑，都可以向表中加载样本数据进行检验。不过，这完全是可以避免的。想象一下如果有 15 或 20 个关系存在这种情况；浪费的时间累积起来就会有很多。

如果遵循这一要素，就完全不必解答这些问题，执行这样的检验。图 10.52 所示为修改后的示意图，其中使用了适宜的外键名称。在这种情形中，外键无疑是合适的。检查这个示意图时，只需随意一瞥，就可以迅速判断表之间的关系类型及其建立的方式。

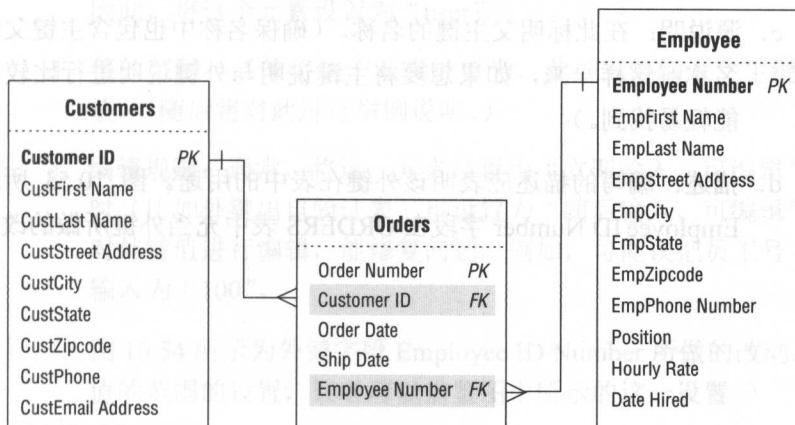


图 10.52 规范的外键

❖ **注意：**当被邀请去分析数据库的特定类型的问题时，笔者常常碰到这种问题。在很多情况下，外键要么是完全不合格，要么显示出严重的数据完整性和关系完整性问题。创建规范的外键（或修改现有外键）后，许多问题就随之消失了。

有次笔者为某个表建立自引用关系时，给外键使用了别名，这是笔者唯一认为合理并且赞同的情况。原因在于，主键和外键在同一表中（多数情况如此），所以每个字段名称必须唯一。

- **使用被复制主键的字段说明副本。**这一点支持理想字段的第六要素。（见第 7 章“即使出现在多个表中，其主要特性始终保持不变。”）不过，外键在通用元素和逻辑元素两个类别的一些设置上都与其父主键存在细微区别。

为外键定义字段说明时，需要修改通用元素中的四个元素。

- a. **说明类型：**因为外键是基于现有主键建立的，它继承了主键说明的副本；所以，将外键说明类型指定为“可复制”。这有助于确保外键说明一致，并提醒你保持这个说明与主键说明同步。
- b. **父表：**写下外键的父表名称。
- c. **源说明：**在此标明父主键的名称。（确保名称中也包含主键父表的名称；这样一来，如果想要将主键说明与外键说明进行比较，就能轻易找到。）
- d. **描述：**编写的描述应表明该外键在表中的用途。图 10.53 所示为 Employee ID Number 字段在 ORDERS 表中充当外键所做的改动。

General Elements

Field Name:	Employee ID Number	SpecificationType:	<input type="checkbox"/> Unique <input type="checkbox"/> Generic <input checked="" type="checkbox"/> Replica
ParentTable:	Orders	Source Specification:	Employee ID Number from the EMPLOYEES table.
Label:	Employee #		
Shared By:			
Alias(es):			
Description:	The identification number of an employee within our organization. The values in this field enable us to identify and keep track of the employees who place orders for our customers.		

图 10.53 ORDERS 表中外键字段 Employee ID Number 的一般元素

另外，还要修改外键字段说明逻辑元素中的五个元素。

- a. 键类型：将这个元素设置为“foreign（外键）”。这是一个相当明显的变化，但是如果不仔细，有时就会把它忽略掉。
- b. 单值性：将这个元素指定为“non-unique”，因为希望能够将单一外键值和父表中的任意数量的记录联系起来。就本文中的示例来说，希望特定员工能管理任意数量的订单。如果将这个元素设定为“unique”，特定员工就只能管理一个订单，这将极大地限制他的销售潜力！（然而，在一对一关系中，会将这个元素指定为“unique”，因为子表中的单一外键只与父表中的一个记录相关联。）
- c. 值的输入者：与父主键不同，由你（或用户）向外键中输入值；因此，将这个元素设置为“user”。
- d. 值的范围：必须将这一元素设置为，此处只能输入父主键的现有值。（随后将对此进行举例说明。）
- e. 编辑规则：通常，将这一元素设置为“立即输入，可编辑”，但有时（比如外键出自验证表）也设置为“随后输入，可编辑”。允许对外键值进行编辑，能修复问题。例如，可能误把员工号“110”输入为“100”。

图 10.54 所示为外键字段 Employee ID Number 所做的改动。（注意值的范围的设置，读者可以借鉴图中所示的这一设置。）

Logical Elements	
Key Type:	<input type="checkbox"/> Non <input type="checkbox"/> Primary <input checked="" type="checkbox"/> Foreign <input type="checkbox"/> Alternate
Key Structure:	<input checked="" type="checkbox"/> Simple <input type="checkbox"/> Composite
Uniqueness:	<input checked="" type="checkbox"/> Non-unique <input type="checkbox"/> Unique
Null Support:	<input type="checkbox"/> Nulls Allowed <input checked="" type="checkbox"/> No Nulls
Values Entered By:	<input checked="" type="checkbox"/> User <input type="checkbox"/> System
Required Value:	<input type="checkbox"/> No <input checked="" type="checkbox"/> Yes
Default Value:	
Range of Values:	Any existing Employee ID Number in the EMPLOYEES table.
Comparisons Allowed:	<input checked="" type="checkbox"/> Same Field <input type="checkbox"/> All <input checked="" type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <= <input type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <= <input checked="" type="checkbox"/> Value Expression <input type="checkbox"/> All <input checked="" type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
Operations Allowed:	<input type="checkbox"/> Same Field <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation <input type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation <input type="checkbox"/> Value Expression <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation

图 10.54 ORDERS 表中外键字段 Employee ID Number 的逻辑元素

为了让读者清晰地认识这些改动的意义，图 10.55 展示出了源说明的逻辑元素。（源说明这一元素存在于一般元素类别中；见图 10.53。）

- 从它所参照的主键中提取值。根据定义，外键的值的范围仅限于父主键的现有值。例如，不能将无效的 Employee ID Number 值输入 ORDERS 表中。输入 ORDERS 表中的任意 Employee ID Number 值都必须在 EMPLOYEES 表中有相同的 Employee ID Number 值存在。这确保了两个表中这两个字段的值保持一致，有助于建立关系层次完整性。

评审每个表中的外键，确保它们都满足外键的要素。如有不符合之处，做出相应修改。如果严格地遵循之前的设计过程，就完全不应该遇到任何问题。

Logical Elements	
Key Type:	<input type="checkbox"/> Non <input checked="" type="checkbox"/> Primary <input type="checkbox"/> Foreign <input type="checkbox"/> Alternate
Key Structure:	<input checked="" type="checkbox"/> Simple <input type="checkbox"/> Composite
Uniqueness:	<input type="checkbox"/> Non-unique <input checked="" type="checkbox"/> Unique
Null Support:	<input type="checkbox"/> Nulls Allowed <input checked="" type="checkbox"/> No Nulls
Values Entered By:	<input type="checkbox"/> User <input checked="" type="checkbox"/> System
Required Value:	<input type="checkbox"/> No <input checked="" type="checkbox"/> Yes
Default Value:	
Range of Values:	1000-9999
Comparisons Allowed:	<input checked="" type="checkbox"/> Same Field <input type="checkbox"/> All <input checked="" type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <= <input type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <= <input checked="" type="checkbox"/> Value Expression <input type="checkbox"/> All <input checked="" type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
Operations Allowed:	<input type="checkbox"/> Same Field <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation <input type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation <input type="checkbox"/> Value Expression <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation

图 10.55 EMPLOYEES 表中主键字段 Employee ID Number 的逻辑元素

建立关系特征

现在, 将为每个关系建立特征。这些特征指示的是删除一个记录所造成的后果, 关系中每个表的参与类型以及参与度。

为每个关系定义删除规则

为关系建立的第一个特征就是**删除规则**。这个规则决定了用户发出请求, 要求删除关系的父表中的特定记录, RDBMS 程序所应做出的反应。删除规则对于关系层次完整性至为重要, 因为它能有效防止孤立记录的出现。所谓孤立记录就是与父表中任何记录都无关联的子表记录。

下列为五种类型的删除规则及规则生效后 RDBMS 的相应反应。

1. 否定: RDBMS 程序不会删除父表中的该记录, 而是保留该记录并将之指定为“不活动”。

2. 限制: 如果子表中存在相关记录, RDBMS 不会删除父表中该记录。必须先让 RDBMS 删除子表中所有相关记录, 然后才能删除父表中该记录。
3. 级联: RDBMS 将采取两个特别行动: 删除父表中的该记录, 它将自动删除子表中所有相关记录。
4. 作废: RDBMS 将删除父表中该记录, 然后将子表中相关记录的外键值更改为 null。如果要运用这一删除规则, 就必须修改该外键字段说明, 将 null 支持这一逻辑元素设置为“允许 null”。
5. 默认设置: RDBMS 将删除父表中的该记录, 然后将子表中相关记录的外键值更改为其外键字段说明中的当前默认值。显然, 为了运用这一规则, 必须先设置好默认值元素。

一般首选“限制”删除规则, 也可根据实际情况使用其他规则。判断哪种删除规则更合适的最佳方法是查看关系示意图。先考虑一下图 10.56。

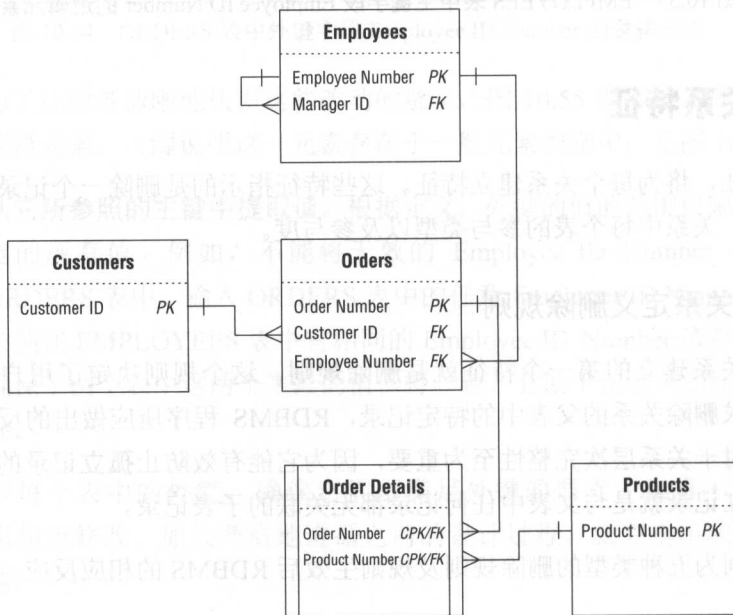


图 10.56 对于给定关系哪种删除规则更合适?

选择一种关系，再看看示意图，思考以下问题：

删除某父表中的一个记录，对应子表中的相关记录应作何改变？

这里提的是一般性问题，以便读者理解背后的前提条件。当对特定关系中的两个表提出这个问题时，可用相应表名称进行替代。比如，处理 EMPLOYEES 和 ORDERS 表之间的关系，就可以提出如下问题：

删除 EMPLOYEES 表中一个记录，ORDERS 表中的相关记录应有何改变呢？

所得到的答案取决于该机构如何使用这些表中的数据，通常也能由此判断应该使用哪种删除规则。

不能删除员工的记录；只能将该员工指定为不活动。（使用否定规则。）

如果存在相关订单记录，就不能删除该员工的记录。（使用限制规则。）

你必须先从 ORDERS 表中删除与该员工相关的订单，然后再从 EMPLOYEES 表中删除该员工。（使用限制规则。）

所有与该员工相关的订单也必须从 ORDERS 表中删除。（使用级联规则。）

所有与该员工相关的订单上的员工编号必须删除。（使用作废规则。）

所有与该员工相关的订单上的员工编号必须重置为主管销售员的员工编号。（使用默认设置规则。）

如果你（或与你合作的人员）无法回答，则将相应关系记录下来，然后继续评审下一关系。等到第 11 章“业务规则”中建立数据库的业务规则时，将重新讨论所有这些关系。现在，不妨假设你收到的是第一个回答，准备使用否定规则。

一旦确认对该关系运用的删除规则，就要将该规则标注在关系示意图中。否定规则使用(D)，限制规则使用(R)，级联规则使用(C)，作废规则使用(N)，默认设置规则使用(S)。将标志标于父表的连接线下。图 10.57 所示即为修改后的 EMPLOYEES 和 ORDERS 表关系示意图。

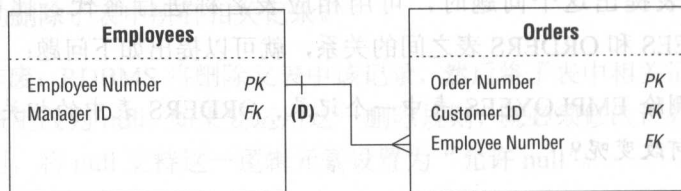


图 10.57 为 EMPLOYEES 和 ORDERS 表之间关系指定否定删除规则

设置删除规则始终是从父表的角度出发，原因在于父表在两个表中更为重要。删除父表中的一个记录，始终会对子表中的相关记录造成一定影响，但是删除子表中的记录则不会对父表中的相关记录造成影响。（在一种特殊情况中，需要为子表建立限制删除规则。详情见第 11 章。）

判断自引用关系的删除规则所运用的问题，与两表间关系所用的问题有一些区别：

删除某父表中的一个记录，其他相关记录的外键值要作何改变呢？

如果处理的是 EMPLOYEES 表的自引用关系，就可以提出如下问题：

删除 EMPLOYEES 表中的一个记录，其他相关记录的外键值要作何改变呢？

同样，其回答通常能揭示应该使用的删除规则：

不能删除当前员工管理者的记录。（使用限制规则。）

如果要删除某管理者记录，就必须先将他负责管理的员工分派给其他管理者。（使用限制规则。）

如果要删除某管理者记录，那么 Manager ID 就必须从他负责管

理的每位员工记录中删除。(使用作废规则。)

如果要删除某管理者记录,那么他负责管理的每位员工记录中的 Manager ID 就必须重置为相应高级管理人员的员工编号。(使用默认设置规则。)

❖注意:之所以不存在级联规则,是因为它根本就不适用于这种关系;你不会仅仅因为管理者离职,就把相应的员工也辞退。在某些情况中,这一规则仍然是一个可行的选择,所以在为其他自引用关系建立删除规则时,应考虑这一规则。

假设得到的是最后一个回答,决定使用默认设置删除规则。于是,在关系示意图中标示出该规则。图 10.58 所示为其结果。

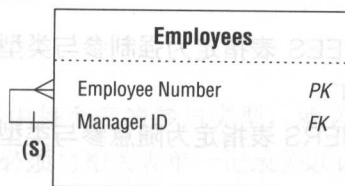


图 10.58 为 EMPLOYEES 表自我参照关系指定默认设置删除规则

识别每个表的参与类型

建立两表之间的关系时,每个表都以特定的方式参与其中。为给定表指定的参与类型,决定了向相关表输入记录之前,该表中是否必须存在一个记录。参与类型分为两种。

1. 强制: 向相关表输入任何记录之前,该表中必须至少存在一个记录。
2. 随意: 向相关表输入记录之前,对该表是否存在记录无要求。

虽然建立关系时,其中每个表的参与类型往往比较明显,能根据常识或某套特定的标准进行判断。但是,大多数表的参与类型一般要等到定义业务规则时才能确定。例如,图 10.59 中 EMPLOYEES 和 CUSTOMERS 表之间的一对多关系。(与图 10.56 中的表略有不同。)

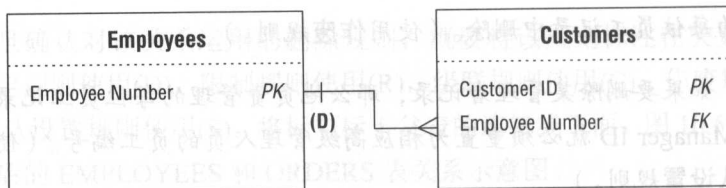


图 10.59 每个表应该指定哪种参与类型呢？

假设必须为每个顾客指派特定员工。由这个员工充当该顾客的销售代表，负责该顾客与公司之间所有交易和通信。尽管每个顾客必须与特定员工相关联，但是有的员工根本不会与顾客打交道。在公司里，许多员工执行的任务都不要求与顾客互动。

这种情况不能归类为特殊情况，但确实表明了该公司开展这部分业务的方式。照此，可以做出如下推断。

- 应该将 EMPLOYEES 表指定为强制参与类型。这能确保至少为给定顾客指派一位员工。
- 应该将 CUSTOMERS 表指定为随意参与类型。这样，你能输入该公司的任意员工。

一旦确定关系中每个表的参与类型之后，就要在关系示意图中将它标示出来。使用垂直线代表强制参与类型，圆圈代表随意参与类型。图 10.60 所示即为调整后 EMPLOYEES 和 CUSTOMERS 表的关系示意图，同时也显示了每个表的参与类型。注意，表示参与类型的符号位于表示关系类型的符号外。

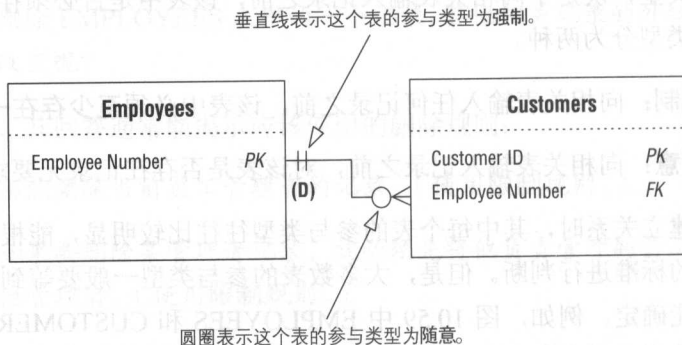


图 10.60 指定 EMPLOYEES 和 CUSTOMERS 表的参与类型

参与类型也适用于自引用关系，但是存在细微的区别。由于自引用关系的本质，应为表中的主键和外键指定参与类型。图 10.61 所示为本章前面的 STAFF 表修改后的关系示意图。



图 10.61 为 STAFF 表的主键和外键指定参与类型

在这种情形中，必须至少有一位具备有效员工编号（主键）的职员充当经理。反过来，无须为新进职员提供经理编号（外键）；这位新进职员也许刚被录用，还未被指派部门或项目。

识别每个表的参与度

既然已经识别关系中每个表的参与类型，就必须也确定每个表的参与度。参与度指示给定表必须与相关表单一记录关联记录的最小数量和容许与相关表单一记录关联记录的最大数量。与判断参与类型一样，判断参与度所依据的因素包括：明显的情况，常识和特定标准。现阶段内一般只能识别一部分表的参与度，等到定义业务规则时再重新考虑剩下的表。

参与度表示为两个数字，数字中间用逗号隔开，外面用圆括号包围。其中第一个数字指示所需相关记录的最小数量，第二个数字指示容许相关记录的最大数量。例如，参与度(2,11)指示该表必须至少存在 2 个、至多不过 11 个与其他表中单一记录相关的记录。

再来看看 EMPLOYEES 和 CUSTOMERS 表。这两表之间存在一对多关系，意味着给定某顾客只能与一个员工相关联，给定某员工则可以与任意数量的顾客相关联。（没错，这一点十分明显。）不过，假设公司刚推出一项新政策，着重抓顾客服务的质量。为了确保每个销售代表按照公司的要求提供服务，这项政策规定每位销售代表同时负责的顾客不得超过 15 个。基于这一情况，可以推断 EMPLOYEES 表的参与度为(1,1)，CUSTOMERS 表的参

与度为(0,15)。

一旦识别了特定表的参与度，就要将该信息在关系示意图标示出来。参与度标示在相应表的连接线上。图 10.62 所示为 EMPLOYEES 和 CUSTOMERS 表修改后的关系示意图。

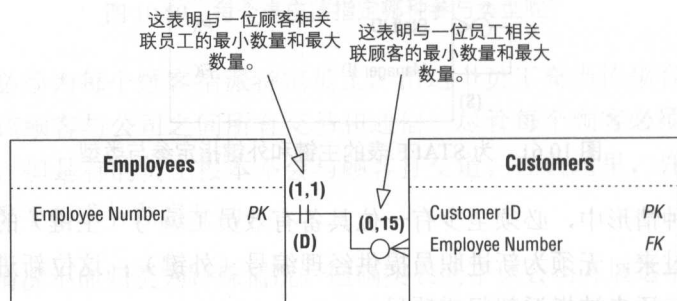


图 10.62 指定 EMPLOYEES 和 CUSTOMERS 表的参与度

参与度也适用于自引用关系，同样，指定参与度的对象是该表的主键和外键。图 10.63 所示为调整后 STAFF 表的示意图，表中包含参与度的信息。

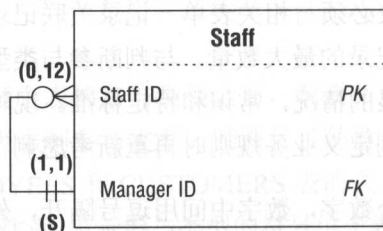


图 10.63 为 STAFF 表的主键和外键指定参与度

Staff ID 的参与度为(0,12)，是因为一位经理最多可以管理 12 个员工；对于尚未指定部门或项目的新经理则还不会分配员工。Manager ID 的参与度为(1,1)，因为给定员工只能受一位经理管理。

若要将表间关系中的任意表或自引用关系中的键字段的参与度指定为无限，第二个数字可以使用“N”表示。例如，图 10.64 中的 ORDERS 表的参与度为无限。应该允许新顾客可以随心所欲地发出任意数量的订单。想想，如果将每个顾客的订单量上限设置为 35，会给公司的业务造成怎样的影响！

如果不能持续地招来新顾客，公司就会因此而破产。

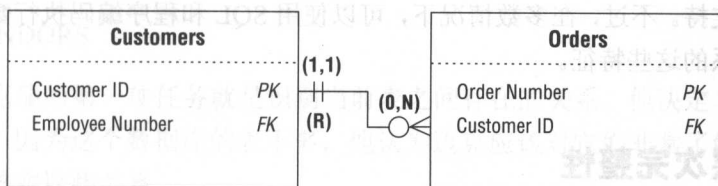


图 10.64 指定 ORDERS 表的参与度为无限

现在，你的任务是为之前建立的每个关系设置关系特征。完成每个表的关系特征后，必须确保对关系示意图进行相应的改动，反映出工作的结果。

与用户和管理人员验证表关系

这一阶段的任务是验证关系。遵照下列检查表，就能更轻松地完成这项任务。

1. 确保正确识别每个关系。
2. 确保正确建立每个关系。
3. 确保每个外键满足外键的要素。
4. 确保为每个关系建立了合适的删除规则。
5. 确保正确识别表间关系的每个表和自引用关系的相应键字段的参与类型。
6. 确保正确识别表间关系的每个表和自引用关系的相应键字段的参与度。

如果所有关系都检查通过，与你合作的所有人都一致认可这些评价，就可以确信所有关系健全合理，可以被应用到视图中。

结语

执行这三种关系特征的难易程度主要取决于所用的 RDBMS 程序。大多

数 RDBMS 并不完全支持所有特征，但是它们都能为删除规则和参与类型提供基础支持。不过，在多数情况下，可以使用 SQL 和程序编码执行数据库中所有关系的这些特征。

关系层次完整性

待核实一个关系建立规范、特征设置合理后，该关系就实现了关系层次完整性。关系层次完整性具备以下优势。

- 关系中两表（或键字段）之间的连接合理健全。使用主键和外键建立一对一或一对多关系以及使用联系表建立多对多关系，就能实现这一点。
- 以一种有意义的方式向每个表中插入新记录。为关系中每个表（或键字段）指定适宜的参与类型，就能保证这一点。
- 删除现有记录，不会带来任何不利影响。为关系指定合适的删除规则，就能确保这一点。
- 合理限制关系中相关记录的数量。为关系中每个表（或键字段）指定适宜的参与度，就能保证这一点。

如你所知，关系层次完整性是整体数据完整性的第三个组成部分。（第一个是表层次完整性，第二个是字段级完整性。）下一章将介绍整体数据完整性的最后一个部分以及为数据库建立业务规则。

案例分析

现在，来识别迈克自行车行的最终表列表中表之间的关系。这部分的设计交给了助手扎克瑞，他现在正在处理下列表：

CUSTOMERS

EMPLOYEES

INVOICES

PRODUCTS

VENDORS

扎克瑞的第一项任务就是识别当前表之间存在的关系。他决定与迈克展开会谈，因为这个数据库的表不多，他认为迈克应该对它们非常了解，能够帮助他核实这些关系。

扎克瑞与迈克会谈前，他创建了一个表矩阵，并尽自己所能识别出了一些关系。图 10.65 所示即为他完成的矩阵。

	Customers	Employees	Invoices	Products	Vendors
Customers			1:N		
Employees			1:N		
Invoices	1:1	1:1		1:N	
Products			1:N		?
Vendors				?	

图 10.65 识别迈克自行车行数据库的表间关系

然后，扎克瑞认真研究了表矩阵，运用合适的公式对表间关系做了判断。以下为他发现的情况。

CUSTOMERS 和 INVOICES 存在一对多关系。

$$(1:1 + 1:N = 1:N)$$

EMPLOYEES 和 INVOICES 存在一对多关系。

$$(1:1 + 1:N = 1:N)$$

PRODUCTS 和 INVOICES 存在多对多关系。

$$(1:N + 1:N = M:N)$$

接着，他画出了这些关系的示意图，将它们放入一个文件夹中，然后带着文件夹前往星巴克咖啡馆与迈克会谈。

在会谈中，迈克和扎克瑞一起验证了这些关系。他们都认定这三种关系正确，然后扎克瑞让迈克看看 PRODUCTS 和 VENDORS 表。他不是很肯定这两表之间的关系，于是他和迈克就此展开了讨论。

扎克瑞：“我想向你了解一下 PRODUCTS 和 VENDORS 表之间的关系。单一产品可以和一个或多个供应商产生联系吗？”

迈克：“嗯，可以这么说。我的意思是一种产品，比如自行车锁，可以与一个或多个供应商相关联。不过，我为每个锁设置了独有的产品编号，将它当作不同的项，而不管供应商是谁。如果你真正问的是，PRODUCTS 表中单一记录是否可以与 VENDORS 表中一个或多个记录相关联，那么答案就是否，因为 PRODUCTS 表中每个记录都只包含 VENDORS 表中一个供应商的引用。”

扎克瑞：“我也这么认为。既然如此，VENDORS 和 PRODUCTS 表之间就存在一对多关系。我本来就想，单一供应商可以和 PRODUCTS 表中多个产品相关联。”

扎克瑞画出了 VENDORS 和 PRODUCTS 表之间一对多关系的关系示意图，接着继续下一步骤。

从父表中复制主键，加入到子表的结构中（其中它作为一个外键），再对关系示意图做出相应的调整。通过这样的过程，他逐一建立了每个一对多关系。图 10.66 所示为修改后的一个示意图。

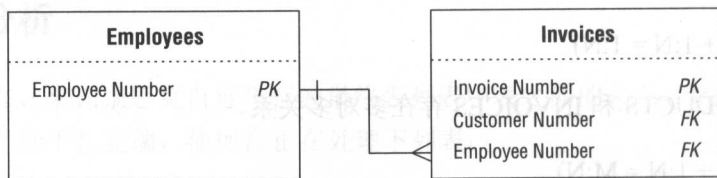


图 10.66 EMPLOYEES 和 INVOICES 表的关系示意图

接着，通过创建一个联系表 INVOICE PRODUCTS，扎克瑞又建立了 INVOICES 和 PRODUCTS 表之间的多对多关系。新联系表是基于 INVOICES

表中的 Invoice Number 字段和 PRODUCTS 表中的 Product Number 字段建立的。图 10.67 所示为 INVOICES 和 PRODUCTS 表修改后的关系示意图。

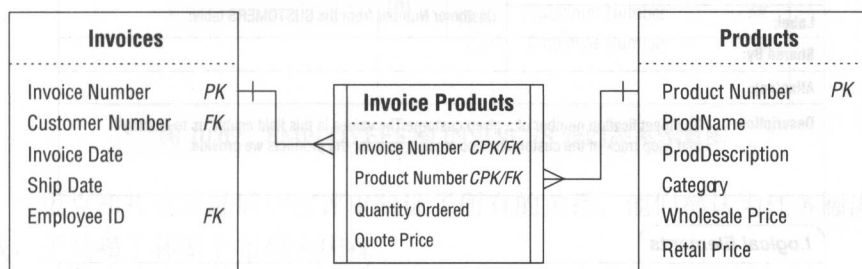


图 10.67 建立 INVOICES 和 PRODUCTS 表之间多对多关系示意图

扎克瑞评审了每个表的结构，以确保满足理想表的要素。可喜的是，他无须进行任何修改，所有表结构都是健全的。然后，他改进每个表中的外键，确保它们都满足外键的要素。最后，扎克瑞修改了每个外键字段说明表中一般元素和逻辑元素部分的若干项。图 10.68 所示即为他修改的一个外键。（图中阴影部分为修改部分。）

General Elements	
Field Name: Customer Number	Specification Type: <input type="checkbox"/> Unique <input type="checkbox"/> Generic <input checked="" type="checkbox"/> Replica
Parent Table: Invoices	Source Specification: Customer Number from the CUSTOMERS table.
Label:	
Shared By:	
Alias(es):	
Description: The identification number of a given customer. The values in this field enable us to identify and keep track of the customers who place orders for the products we provide.	

Logical Elements	
Key Type: <input type="checkbox"/> Non <input type="checkbox"/> Primary <input checked="" type="checkbox"/> Foreign <input type="checkbox"/> Alternate	Edit Rule: <input checked="" type="checkbox"/> Enter Now, Edits Allowed <input type="checkbox"/> Enter Now, Edits Not Allowed <input type="checkbox"/> Enter Later, Edits Allowed <input type="checkbox"/> Enter Later, Edits Not Allowed <input type="checkbox"/> Not Determined At This Time
Key Structure: <input checked="" type="checkbox"/> Simple <input type="checkbox"/> Composite	
Uniqueness: <input checked="" type="checkbox"/> Non-unique <input type="checkbox"/> Unique	
Null Support: <input type="checkbox"/> Nulls Allowed <input checked="" type="checkbox"/> No Nulls	
Values Entered By: <input checked="" type="checkbox"/> User <input type="checkbox"/> System	
Required Value: <input type="checkbox"/> No <input checked="" type="checkbox"/> Yes	
Default Value:	
Range of Values: Any existing Customer Number in the CUSTOMERS table.	
Comparisons Allowed:	
<input checked="" type="checkbox"/> Same Field <input type="checkbox"/> All <input checked="" type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> < <input type="checkbox"/> <=	
<input type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> < <input type="checkbox"/> <=	
<input checked="" type="checkbox"/> Value Expression <input type="checkbox"/> All <input checked="" type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> < <input type="checkbox"/> <=	
Operations Allowed:	
<input type="checkbox"/> Same Field <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> Concatenation	
<input type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> Concatenation	
<input type="checkbox"/> Value Expression <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> Concatenation	

图 10.68 INVOICES 表中外键字段 Customer ID 的字段说明表中一般元素和逻辑元素部分

扎克瑞的下一任务是为每个表建立合适的关系特征。他从定义删除规则入手，再识别关系中每个表的参与类型和程度。最后，他将这些特征标示在关系示意图中，也就完成了这个任务。图 10.69 所示为其中一个完成的示意图。

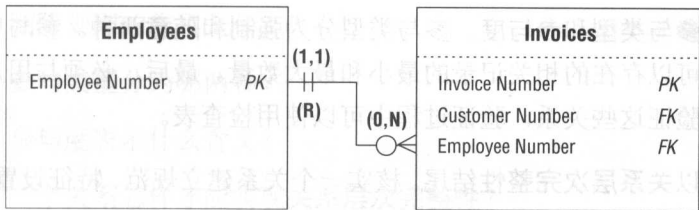


图 10.69 EMPLOYEES 和 INVOICES 表的关系示意图

迈克和扎克瑞最后一次评审验证了所有的关系。他们都认为任务圆满完成，于是喝了杯摩卡布列夫庆祝。

小结

本章开篇讨论了特定两表之间存在的三种关系，包括一对一、一对多和多对多。其中一对多关系最为常见，多对多关系则会引发问题，且必须予以解决。然后，又介绍了自引用关系，自引用关系存在于给定表内的字段之间。与表间关系相似，自引用关系也分为一对一、一对多和多对多三种。

接下来，讨论了如何识别表间关系。首先，建立和使用表矩阵，然后运用关联型和情境型问题帮助识别给定关系。接着，还介绍了三个公式，用于判断两表之间和自引用关系中记录之间的真正关系。

随后，本章继续讨论如何建立关系。建立一对一和一对多关系要使用主键和外键，建立多对多关系则要使用联系表。接着，我们简单地回顾了多值字段，介绍了如何正确使用一对多关系更为高效地解决多值字段。建立自引用关系与建立表间关系十分相似。建立好关系之后，必须评审所有表结构，确保它们仍然满足理想表的要素。

下一个讨论的主题是外键。每个外键必须符合外键的要素。外键与父主键共享相同的名称，这一点十分重要。另外，还必须修改充当外键字段的字段说明中某些元素，外键的值也必须取自父主键。

然后，讨论了关系特征。首先介绍了如何为关系定义删除规则，删除规则分为四种。接着，介绍了如何识别表间关系中每个表和自引用关系中每个

键字段的**参与类型**和**参与度**。参与类型分为**强制**和**随意**两种。参与度指示给定关系中可能存在的相关记录的最小和最大数量。最后，必须与用户和管理人员一起验证这些关系，验证过程中可以使用检查表。

本章以**关系层次完整性**结尾。核实一个关系建立规范、特征设置合理后，该关系就实现了关系层次完整性。

思考题

1. 说说关系重要的两个原因。
2. 说出三种关系的名称。
3. 哪种关系引发的问题最多？
4. 说说处理多对多关系可能遇到的两个问题。
5. 什么是自引用关系？
6. 识别数据库中表间关系应从何处入手？
7. 哪两种问题能帮助你识别现有关系？
8. 在表矩阵中，一对多关系使用什么缩写符号表示？
9. 如何判断矩阵中两表之间的正式关系类型？
10. 如何建立一对多关系？
11. 判断题：如果一个表具有自引用关系，从该表中检索信息就将变得烦琐且有些困难。
12. 如何建立多对多自引用关系？
13. 如何改进外键？
14. 外键的字段说明中哪两个元素类别必须修改？

15. 删除规则的作用是什么?
16. 参与类型分为哪两种?
17. 参与度表示什么含义?
18. 一个关系怎样才能实现关系层次完整性?

数据库的参与类型和数量，它们与什么数据相关联，数据如何被创建、更新、删除和查询。数据库设计者必须验证这些关系，验证过程中可以使用检查表。

第 11 章

业务规则

思考题

你会因你打破的规则而被人铭记。

——道格拉斯·麦克阿瑟将军

本章内容

什么是业务规则

业务规则的分类

定义和建立业务规则

验证表

评审业务规则规范表

案例分析

小结

思考题

纵观数据库设计过程，已经通过执行各项任务，建立了各个层次的数据完整性。到目前为止，已经建立了表层次完整性、字段级完整性和关系层次完整性。这样一来，就已经确保了表和字段结构合理健全，输入字段的数据一致且基本有效，以及关系规范且有意义。本章将介绍如何建立整体数据完整性的最后一个部分：业务规则。

什么是业务规则

业务规则表示对数据库特定方面实施的某种形式的限制，比如特定字段说明中的元素和给定关系的特征。业务规则基于机构认知和使用其数据的方式建立，机构认知和使用其数据的方式则是从它运作或开展业务的方式中得出。

选择是任何设计过程的一个重要方面。例如，在数据库设计中，必须选择数据库中存储哪些数据；并非机构可能使用的所有数据都需要存储。最终选择存储的数据及其存储方式将由机构使用数据的方式决定。一家医院可能希望将各种事件的时间精确到秒，而一个仓库只需要记录任意事件的日期。

为了指导数据库设计过程（以及后续 RDBMS 中实现数据库过程）所需做出的这些以及其他选择，需要制定一份正式的机构业务规则。这些规则将影响到数据库的方方面面，比如所收集和存储的数据，定义和建立关系的方式，数据库提供的信息类型，以及数据本身的安全和保密。想要设计一套业务规则同时适用于多个机构的业务规则几乎是不可能的。每个机构都有自己的数据和信息要求，每个机构都有各自开展业务的独特方式；因此，每个机构都需要一套专属的业务规则。

下列为一个典型的业务规则的示例：

对于任意订单，Ship Date 不得早于 Order Date。

这条业务规则对 Ship Date 字段说明的值的范围元素提出了限制。它有助于确保 Ship Date 的值在销售订单中合理规范。如果没有这一限制，就可以向该字段输入任意日期（包括早于 Order Date 的日期），这样 Ship Date 字段的值就完全无意义。正是由于这条业务规则，才能保证 Ship Date 字段的值有意义。

因为业务规则取决于机构认知和使用其数据的方式，所以极有可能特定规则被多个机构使用，而原因却完全不同。

例如，贝尔艾尔高中的音乐技艺因其培养的学生音乐家才能出众远近闻名。这里的学生之所以能达到高水平的音乐才能，是因为学校鼓励他们专攻音乐学习，将所学的乐器限制在不超过两种。在城镇的另一区域，湖城高中（私立学校）的音乐科也培养学生音乐家较高的音乐才能，学校也同样帮助学生专攻音乐学习。不过，这里的学生学习乐器不超过两种则是由于学校政策；这所学校乐器库存非常有限。

无独有偶，两所学校都在设计自己的数据库。两所学校都要运用数据库支持其日常运作和行政职能。凑巧的是，两个数据库中都包含了图 11.1 所示的表。

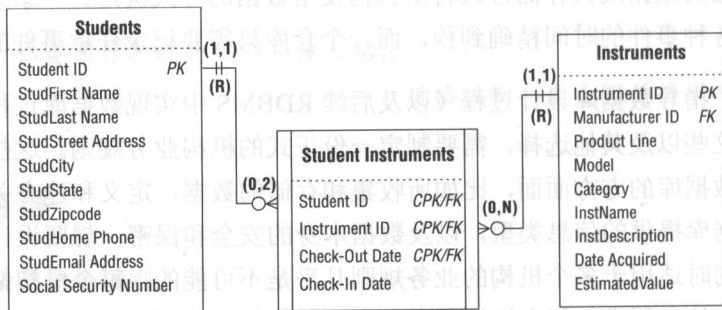


图 11.1 贝尔艾尔高中和湖城高中数据库中的表

两所学校都处在数据库设计的相同阶段，当前都在建立业务规则。结果，两所学校都在各自的数据库中使用下列业务规则：

一名学生同时考核的乐器不得超过两种。

这条业务规则适用于 STUDENTS 表和 STUDENT INSTRUMENTS 表之间的参与度。这样，STUDENTS 中单一记录和 STUDENT INSTRUMENTS 表中相关联的记录就不能超过两个。STUDENT INSTRUMENTS 表中每个记录的 Check-In Date 值为 null；Check-In Date 字段的 null 值表示该学生仍然占有该乐器。

这条规则确实同时适用于两所学校，但是每所学校有着各自不同的考虑。贝尔艾尔高中需要这一规则，是由于其音乐课程建立的方式，而湖城高

中则是因为其乐器库存有限。两所学校开发一条相同的规则完全是巧合。这个例子表明了业务规则的确是依据机构运作或开展业务的方式，同时它也说明了每个机构必须拥有自己独有业务规则的原因。

另外，这个例子也揭示了另一问题：在数据库的逻辑设计中，无法建立特定业务规则所规定的限制，比如上例所述。例如，这里没有清晰地表明，为了判断一名学生是否能借出另一种乐器，必须测试 Check-In Date 值。必须解决和建立数据库逻辑设计之外的限制。怎样判断在这一过程中给定限制是否表述正确呢？可以通过识别其业务规则类型做出判断。

业务规则类型

业务规则主要分为两种：**面向数据库**和**面向应用程序**。这两种业务规则都实施某种形式的限制且有效维持整体数据完整性，但是它们的区别在于所建立的地点和方式。

面向数据库业务规则规定的限制，可以在数据库的逻辑设计中建立。通过修改各个字段说明元素、关系特征或同时修改这两者，就可以实施给定限制。这些表述限制的语句就是面向数据库业务规则。规则语句必须清楚准确。例如，有一个 VENDORS 表，为该表中 VendState 字段定义的业务规则如下：

我们专门与来自太平洋西北地区的供应商开展业务。

这条业务规则将输入字段的 VendState 值限制为 WA、OR、ID 和 MT。修改 VendState 字段说明中值的范围元素，就能建立这条业务规则的限制。图 11.2 所示为所做的修改。

面向应用程序业务规则规定的限制，不能在数据库的逻辑设计中建立，而是必须在数据库的物理设计或数据库应用程序的设计中建立。在后两个环境中，这些限制更为有效、更具意义。（本书使用数据库应用程序一词代指 RDBMS 软件中编写的程序，它便于机构人员使用数据库和执行日常工作相关的任务。）

Logical Elements	
Key Type:	<input checked="" type="checkbox"/> Non <input type="checkbox"/> Primary <input type="checkbox"/> Foreign <input type="checkbox"/> Alternate
Key Structure:	<input type="checkbox"/> Simple <input type="checkbox"/> Composite
Uniqueness:	<input checked="" type="checkbox"/> Non-unique <input type="checkbox"/> Unique
Null Support:	<input type="checkbox"/> Nulls Allowed <input checked="" type="checkbox"/> No Nulls
Values Entered By:	<input checked="" type="checkbox"/> User <input type="checkbox"/> System
Required Value:	<input type="checkbox"/> No <input checked="" type="checkbox"/> Yes
Default Value:	None
Range of Values:	ID, MT, OR, WA
Comparisons Allowed:	
<input checked="" type="checkbox"/> Same Field	<input type="checkbox"/> All <input checked="" type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
<input type="checkbox"/> Value Expression	<input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
Operations Allowed:	
<input type="checkbox"/> Same Field	<input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation
<input type="checkbox"/> Value Expression	<input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation

图 11.2 实施面向数据库业务规则所规定的限制

下列为典型面向应用程序的业务规则的一个示例：

拥有“优先”级别的顾客的所有采购均享受 15% 的折扣。

这条业务规则是基于特定级别决定顾客所有采购享受的折扣。这个限制不能建立在逻辑设计中，出于两个原因：无存储折扣量的字段（折扣量是计算的结果，表中不允许出现计算字段），以及无法显示折扣所依据的标准（顾客的级别）。因此，这条规则必须建立在数据库的物理设计或数据库应用程序设计中。

❖ 注意：讨论实际定义和建立面向应用程序业务规则的方式超出了本书的范围。一些 RDBMS 所提供的工具，让你能够相对轻松地执行常见的应用程序业务规则；大多数 RDBMS 则需要编写程序编码，才能实现和实施这些规则。

尽管这两种业务规则都很重要，但是这一阶段将只聚焦于面向数据库的业务规则。

❖注意：本书的剩余部分直接将面向数据库业务规则称为业务规则。

业务规则的分类

为便于理解和定义，本书将业务规则分为两类：字段特有和关系特有。

字段特有业务规则

这一类别的业务规则，施加限制的对象是特定字段说明中的元素。给定规则所影响的元素数量取决于定义该规则的方式。例如，如下规则只影响一个元素：

订单日期表示为长形式，比如“January 10, 2012 (2012 年 1 月 10 日)”。

这条规则影响到了 ORDERS 表中 Order Date 字段的显示格式元素。建立这条规则就要修改 Order Date 字段说明的显示格式元素，表明日期应该显示的格式。

下列规则则影响多个元素：

我们必须能为加拿大籍顾客存储邮编。

这条规则影响到 CUSTOMERS 表中 CustZipcode 字段说明的数据类型、字符支持和显示格式三个元素。加拿大的邮编包含字母，所以为了实施这条规则定义的限制，必须做出如下修改。

1. 将数据类型的设置修改为“字母数字”。
2. 字符支持元素下勾选“字母”。
3. 修改显示格式元素，确保加拿大邮编中的字母可以大写。

图 11.3 所示为 CustZipcode 字段说明修改后的物理元素部分。

Physical Elements		
Data Type:	Alphanumeric	Character Support: <input checked="" type="checkbox"/> Letters (A-Z) <input type="checkbox"/> Keyboard (., / \$ # %) <input checked="" type="checkbox"/> Numbers (0-9) <input type="checkbox"/> Special (© ® ™ Σ π)
Length:	6	
Decimal Places:	Not Applicable	
Input Mask:	Not Applicable	
Display Format:	Uppercase letters where applicable.	

图 11.3 建立 CustZipcode 的字段特有业务规则

关系特有业务规则

这种业务规则施加的限制会影响到关系的特征。例如，假设正在处理图 11.4 所示的表和关系。

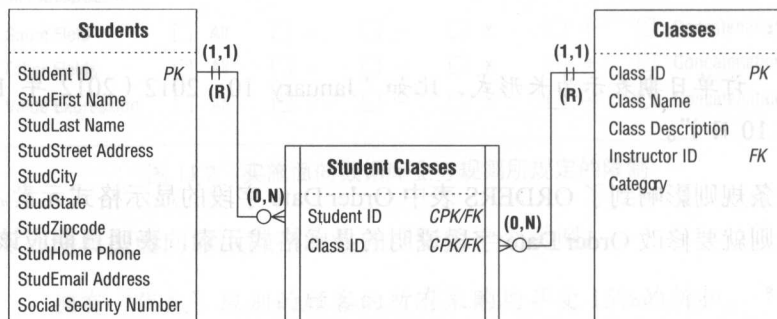


图 11.4 某学校数据库中的表和关系

你认为必须限制参加每门课程的学生人数。于是，定义了以下规则：

每门课必须最少有 5 个学生，最多不超过 20 个学生。

这条业务规则影响到了 CLASSES 和 STUDENT CLASSES 表之间关系的参与度。通过修改关系示意图，标示出 STUDENT CLASSES 表中与 CLASSES 表中单一记录相关联的记录必须至少有 5 个至多不超过 20 个，由此实施了这条规则定义的限制。（根据判断，也可以从这条业务规则中推断出 STUDENT CLASSES 表的参与类型为强制。当且仅当某课程选修的学生数达 5 人时，才能在 CLASSES 表中输入新课程或保留已有课程。）图 11.5 所示为

建立这条业务规则所做出的改动。

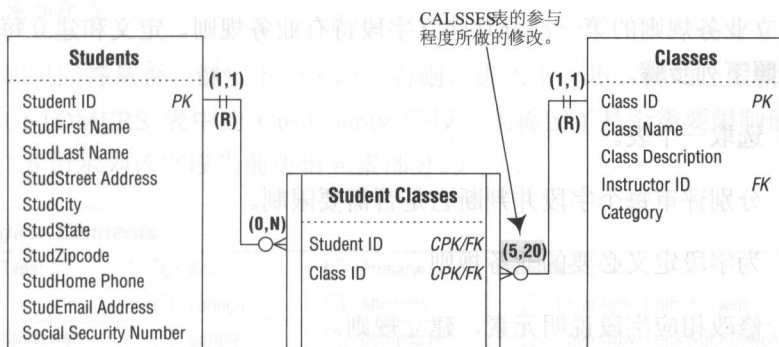


图 11.5 建立关系特有业务规则

定义和建立业务规则

这一阶段的任务为定义和建立业务规则。记住，业务规则必须基于机构认知和使用其数据的方式。众所周知，机构认知和使用其数据的方式取决于它运作和开展业务的方式。完成本阶段任务最佳的途径就是，先定义和建立字段特有业务规则，后定义和建立关系特有业务规则。这一途径将有助于保持对所定义规则类型的关注，也能避免在两种业务规则之间摇摆。否则，往往会导致混淆、甚至失败。

与用户和管理人员合作

你将与用户和管理人员中的代表再次合作。安排好会谈议程，和他们一道定义和建立合适的业务规则。按照一个整体的模式团结协作，就能确保业务规则所施加的限制切实有效，避免造成混淆或产生歧义。整体中的任何人（包括你自己）对某限制持有疑问，就可以讨论它将对相关字段或关系造成的影响以及施加该限制的利弊。然后，再根据讨论的结果，决定保留该规则还是完全把它排除。

定义和建立字段特有业务规则

建立业务规则的第一部分是建立字段特有业务规则。定义和建立每条规则可按照下列步骤。

1. 选取一个表。
2. 分别评审每个字段并判断它是否需要限制。
3. 为字段定义必要的业务规则。
4. 修改相应字段说明元素，建立规则。
5. 选定测试该规则的操作。
6. 将该规则记录在业务规则规范表中。

下面详细介绍每个步骤。

第 1 步：选取一个表

开始可任意选取一个表，因为最终将会把上述过程运用到数据库的每个表。不过，最好先选取一个结构较为熟悉的表，这便于快速适应上述过程，也将有利于后续部分处理包含复杂字段的表。

然后，想想该表所表示的主题，再提出以下问题：

机构如何使用基于该主题或与之相关的信息呢？

该表本身或与数据库中其他表有什么关系吗？

如有必要，可查询最终表列表并参阅该表的描述，以及查看包含该表的所有关系示意图。这些信息无论是为下一步做准备还是为该表定义规则都非常有用。

第 2 步：分别评审每个字段并判断它是否需要限制

检查每个字段的字段说明表，判断是否应该对其中的元素运用限制。评审给定说明表的同时，思考第 1 步中提出的问题，再提出以下问题：

基于该表在数据库中的用途，是否有必要对这一说明中的元素施加限制？

如果回答是否，继续下一字段；否则，进入下一步。例如，假设正在处理 CUSTOMERS 表中的 CustCounty 字段，刚提出了是否需要限制的问题。（图 11.6 所示为该字段当前逻辑元素部分。）

Logical Elements	
Key Type:	<input checked="" type="checkbox"/> Non <input type="checkbox"/> Primary <input type="checkbox"/> Foreign <input type="checkbox"/> Alternate
Key Structure:	<input type="checkbox"/> Simple <input type="checkbox"/> Composite
Uniqueness:	<input checked="" type="checkbox"/> Non-unique <input type="checkbox"/> Unique
Null Support:	<input checked="" type="checkbox"/> Nulls Allowed <input type="checkbox"/> No Nulls
Values Entered By:	<input checked="" type="checkbox"/> User <input type="checkbox"/> System
Required Value:	<input checked="" type="checkbox"/> No <input type="checkbox"/> Yes
Default Value:	None
Range of Values:	King, Kitsap
Comparisons Allowed: <input type="checkbox"/> Same Field <input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <= <input type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <= <input checked="" type="checkbox"/> Value Expression <input type="checkbox"/> All <input checked="" type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=	
Operations Allowed: <input type="checkbox"/> Same Field <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation <input checked="" type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input checked="" type="checkbox"/> Concatenation <input checked="" type="checkbox"/> Value Expression <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input checked="" type="checkbox"/> Concatenation	

图 11.6 CustCounty 字段当前的逻辑元素设置

如果收到如下回答，应该进入下一步。

“嗯，上头希望把顾客信息具体到县，所以我们必须确保每位顾客记录的信息细化到县名。实际上，我们才将皮尔斯县和斯诺霍米什县纳入销售区域，所以记录这两个县名非常重要。”

这一回答明显表达了肯定的意思，所以你将进入下一步，继续为这个字段定义业务规则。

第 3 步：为字段定义必要的业务规则

识别第 2 步的回答中暗含的限制，为 CustCounty 字段定义合适的业务规则。然后将每个限制转化为一条规则。

第 2 步中的回答暗示了应该施加在 CustCounty 字段上的两个可能的限制：每个顾客都要求记录所在县名。这个字段的值的范围限于四个特定县（当前字段说明上有两个，回答中暗含了两个新县名。）将这些限制转化为业务规则时，可能运用如下两个语句：

每个顾客必须与一个县相关联。

可输入该字段的县名包括金、基特萨普、皮尔斯和斯诺霍米什。

定义完合适的业务规则后，就可以进入第 4 步。

第 4 步：修改相应字段说明元素，建立规则

修改相应的字段说明表中的元素，建立第 3 步定义的所有业务规则。（记住，一些规则可能影响多个元素。）不过，首先必须识别规则影响到的字段说明中的元素。例如，第 3 步定义的 CustCounty 字段的第一条业务规则：

每个顾客必须与一个县相关联。

据此可以推断，这条规则影响到了要求值、null 支持以及编辑规则三个元素。然后，这些元素进行合适的修改。在这一情形中，将要求值设置为“是”，null 支持设置为“禁用 null”，编辑规则设置为“立即输入，可编辑”。

从上可知，为了判断影响到的字段说明元素，必须仔细检查每条业务规则。随着建立业务规则熟练度的提高，确定元素将变得更加容易。

现在来看看第二条业务规则：

可输入该字段的县名包括金、基特萨普、皮尔斯和斯诺霍米什。

这条业务规则影响到值的范围，于是将它的设置修改为“金、基特萨普、皮尔斯和斯诺霍米什”。图 11.7 所示为 CustCounty 字段说明表修改后的逻辑

元素部分。

Logical Elements	
Key Type:	<input checked="" type="checkbox"/> Non <input type="checkbox"/> Primary <input type="checkbox"/> Foreign <input type="checkbox"/> Alternate
Key Structure:	<input type="checkbox"/> Simple <input type="checkbox"/> Composite
Uniqueness:	<input checked="" type="checkbox"/> Non-unique <input type="checkbox"/> Unique
Null Support:	<input type="checkbox"/> Nulls Allowed <input checked="" type="checkbox"/> No Nulls
Values Entered By:	<input checked="" type="checkbox"/> User <input type="checkbox"/> System
Required Value:	<input type="checkbox"/> No <input checked="" type="checkbox"/> Yes
Default Value:	None
Range of Values:	King, Kitsap, Pierce, Snohomish
Comparisons Allowed:	
<input type="checkbox"/> Same Field	<input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
<input checked="" type="checkbox"/> Value Expression	<input type="checkbox"/> All <input checked="" type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=
Operations Allowed:	
<input type="checkbox"/> Same Field	<input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation
<input checked="" type="checkbox"/> Other Fields	<input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input checked="" type="checkbox"/> Concatenation
<input checked="" type="checkbox"/> Value Expression	<input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input checked="" type="checkbox"/> Concatenation

图 11.7 CustCounty 字段逻辑元素部分修改后的设置

第 5 步：选定测试该规则的操作

测试业务规则所施加的限制可借助三种操作：向相应表中插入一个记录或向字段中插入一个条目；删除相应表中的一个记录或字段中的一个值；更新一个字段值。既然已经建立业务规则并且理解了它所施加的限制，就可以通过确认违反该规则的条件，决定测试该规则的操作。思考以下问题能使这项任务变得相对简单。

假如向这个表输入一个新记录，会违反这条规则吗？

假如不向这个表输入一个新记录，会违反这条规则吗？

假如删除这个表中的一条记录，会违反这条规则吗？

假如向这个字段中输入一个值，会违反这条规则吗？

假如更新这个字段的值，会违反这条规则吗？

假如删除这个字段的值，会违反这条规则吗？

一旦认定哪种操作会引发违反这条规则，就记录下来；在下一步中将用到它们。在 RDBMS 中实现数据库时，这些信息也有助于以最为有效的方式建立这条规则。

在这一情况中，当试图向 CustCounty 字段中插入一个值时，就将测试 CustCounty 字段的业务规则，因为该值必须处于一定范围。当试图删除该字段中的一个值时，也将测试这条规则，因为该值不能为 null。

第 6 步：将该规则记录在业务规则规范表中

填写业务规则规范表，为给定业务规则建立文档，供以后参考。无论规则属于哪种类型，都应该归入文档。业务规则规范表具备以下三个优点。

1. 可以将所有面向数据库的业务规则使用文档记录。这能有效确保每条规则定义合理、建立规范。
2. 可以将所有面向应用程序的业务规则使用文档记录。尽管无法在数据库逻辑设计过程中建立这种规则，但是至少能指示其基本元素。等到 RDBMS 中实现数据库或创建该数据库的应用程序时，使用文档记录这种业务规则的信息将会显示出无可估量的价值。
3. 为记录所有业务规则提供一种标准方法。以一致规范的方式记录业务规则，更易于保存和维护。使用统一的格式也便于排除业务规则中的故障；业务规则的所有细节都将显示在规范表中。

业务规则规范表包含下列项。

- **陈述 (statement)**：描述业务规则本身的文本。应做到简明扼要，准确表达必要的限制，无任何含糊不清或歧义之处。下列为规范陈述的示例：

一名经纪人所负责的艺人不得超过 25 人。

- **限制 (constraint)**: 简单说明该限制如何运用到相应表或字段。例如, 前面例子中的业务规则所施加的限制可进行如下表述:

AGENTS 表中的单一记录与 ENTERTAINERS 表中相关联的记录不得超过 25 个。

- **类型 (type)**: 指示该规则属于面向数据库或面向应用程序类型。
- **分类 (category)**: 指示该规则属于字段特有或关系特有。
- **测试方法 (test on)**: 指示测试该业务规则所施加限制的方法 (插入、删除、更新)。
- **受影响结构 (structures affected)**: 根据业务规则的类型, 该限制将影响某个字段或关系。此处指示该规则影响的字段名称或受影响关系中涉及的表名称。
- **受影响字段元素 (field elements affected)**: 字段特有业务规则会影响到该字段说明的一个或多个元素。此处指示受该规则影响的元素。
- **受影响关系特征 (relationship characteristics affected)**: 关系特有业务规则会影响到该关系的一个或多个特征。此处指示受该规则影响的特征。
- **采取的措施 (action taken)**: 此处指示对字段说明的元素或关系示意图所做的修改。叙述语句必须尽可能简单明了。如实施该业务规则引发问题, 此处语句则作为建立该规则所采取步骤的准确陈述, 用以检查步骤是否完全落实以及该规则的建立是否规范。

现在, 填写第 4 步中建立的业务规则规范表。图 11.8 所示为记录 CustCounty 字段业务规则的完整业务规则规范表。

BUSINESS RULE SPECIFICATIONS		
Rule Information		
Statement: A county must be associated with each customer		
Constraint: An entry must be made into the CustCounty field; it cannot be Null.		
Type: <input checked="" type="checkbox"/> Database Oriented <input type="checkbox"/> Application Oriented	Category: <input checked="" type="checkbox"/> Field Specific <input type="checkbox"/> Relationship Specific	Test On: <input checked="" type="checkbox"/> Insert <input type="checkbox"/> Update <input checked="" type="checkbox"/> Delete
Structures Affected		
Field Names: CUSTCOUNTY		
Table Names:		
Field Elements Affected		
Physical Elements		
<input type="checkbox"/> Data Type	<input type="checkbox"/> Decimal Places	<input type="checkbox"/> Input Mask
<input type="checkbox"/> Length	<input type="checkbox"/> Character Support	<input type="checkbox"/> Display Format
Logical Elements		
<input type="checkbox"/> Key Type	<input type="checkbox"/> Values Entered By	<input type="checkbox"/> Comparisons Allowed
<input type="checkbox"/> Key Structure	<input checked="" type="checkbox"/> Required Value	<input type="checkbox"/> Operations Allowed
<input type="checkbox"/> Uniqueness	<input type="checkbox"/> Default Value	<input checked="" type="checkbox"/> Edit Rule
<input checked="" type="checkbox"/> Null Support	<input type="checkbox"/> Range of Values	
Relationship Characteristics Affected		
<input type="checkbox"/> Deletion Rule	<input type="checkbox"/> Type of Participation	<input type="checkbox"/> Degree of Participation
Action Taken		
Required Value was set to "Yes," Null Support was set to "No Nulls," and Edit Rule was set to "Enter Now, Edits Allowed."		

图 11.8 业务规则规范表示例

定义和建立关系特有业务规则

定义和建立字段特有业务规则之后，第二部分就是处理关系特有业务规则。完成这一任务包括以下步骤。

1. 选取一个关系。
2. 评审该关系并判断它是否需要限制。
3. 为该关系定义必要的业务规则。
4. 修改相应关系特征，建立规则。
5. 选定测试该规则的操作。
6. 将该规则记录在业务规则规范表中。

从上可知，这些步骤和字段特有业务规则所用的步骤相似。现在，来进一步看看这些步骤吧。

❖注意：上述步骤同时适用于自引用关系和表间关系。不过，本文后面的讨论主要基于表间关系，因为表间关系更为常见。

第1步：选取一个关系

最先选取哪个关系并不重要，因为所有关系都将经历这个过程。选定关系后，评审其关系示意图。然后，思考该关系涉及的表表示什么以及它们为何相关联，可以提出下列问题：

这些表提供什么样的信息？

这两个表之间的关系有何重要之处？

以上两个问题的答案将有助于为该关系定义必要的业务规则，思考这些问题为下一步做准备。

第2步：评审该关系并判断它是否需要限制

简单评审每个关系特征，记住其当前设置。然后将关系当作一个整体进

行检查,判断它是否需要某种限制。评审关系的同时,记住第 1 步中所提问题的答案。可借助以下问题帮助判断是否需要限制:

根据机构运作和开展业务的方式,需要对这个关系施加某种限制吗?

如果得到肯定回答,就可以进入下一步;否则,评审下一关系,对其重复这一步骤。例如,假设在为一家小型的舞蹈室设计数据库,正好要处理图 11.9 中 INSTRUCTORS 和 INSTRUCTOR CLASSES 表之间的关系。

Logical Elements	
Key Type:	<input type="checkbox"/> Non <input type="checkbox"/> Primary <input checked="" type="checkbox"/> Foreign <input type="checkbox"/> Alternate
Key Structure:	<input checked="" type="checkbox"/> Simple <input type="checkbox"/> Composite
Uniqueness:	<input checked="" type="checkbox"/> Non-unique <input type="checkbox"/> Unique
Null Support:	<input type="checkbox"/> Nulls Allowed <input checked="" type="checkbox"/> No Nulls
Values Entered By:	<input checked="" type="checkbox"/> User <input type="checkbox"/> System
Required Value:	<input type="checkbox"/> No <input checked="" type="checkbox"/> Yes
Default Value:	None
Range of Values: Any value within the Category ID field in the CATEGORIES table	
Comparisons Allowed: <input checked="" type="checkbox"/> Same Field <input type="checkbox"/> All <input checked="" type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <= <input type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <= <input checked="" type="checkbox"/> Value Expression <input type="checkbox"/> All <input checked="" type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=	
Operations Allowed: <input type="checkbox"/> Same Field <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation <input type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation <input type="checkbox"/> Value Expression <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation	

图 11.9 某舞蹈室数据库中表间关系示意图

现在,借助如下问题有助于判断该关系是否需要限制:

根据这家舞蹈室运作和开展业务的方式,需要对这个关系施加某种限制吗?

如果你得到类似下列回答,可继续进入下一步:

是的。我们要求所有教员教授至少一门课程。不过,每位教员教授的课程至多不能超过 8 门。

下一步将使用这一回答作为一条业务规则的基础。

第 3 步：为该关系定义必要的业务规则

接下来，根据第 2 步中的回答定义合适的业务规则。识别回答中隐含的限制，并将它转化为业务规则。例如，可以从该回答中推断出两个限制：一个教员可教授的最少课程数量为 1，最多课程数量为 8。这两个限制转化为一条业务规则，可陈述如下：

一位教员必须教授一门课程，且至多不超过 8 门课程。

定义完规则后，继续下一步。

第 4 步：修改相应关系特征，建立规则

修改该关系示意图中相应特征，建立刚才所定义的业务规则。修改之前，再次考虑一下该业务规则的陈述，识别受该规则影响的关系特征：

一位教员必须教授一门课程，且至多不超过 8 门课程。

该规则所施加的限制影响到一位教员可教授的课程数量，所以要修改 INSTRUCTOR CLASSES 表的参与度，将它设置为“(1,8)”。该规则也影响了 INSTRUCTOR CLASSES 表的参与类型。必须将其参与类型的设置修改为“强制”，因为 INSTRUCTORS 表中单一记录必须至少与 INSTRUCTOR CLASSES 表中一个记录相关联。图 11.10 所示为修改后的关系示意图。

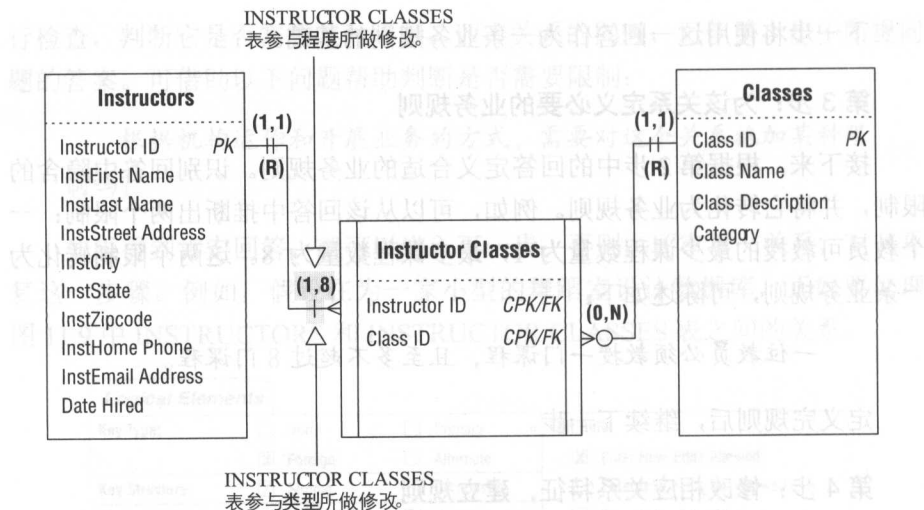


图 11.10 修改后的关系示意图

第 5 步：选定测试该规则的操作

前面提到，插入、删除或更新表记录或字段值时，就会测试业务规则所施加的限制。既然已经建立业务规则且知道它对该关系的影响，就可以通过识别违反该规则的条件，判断采取什么样的操作进行测试。下列问题能帮助你做出判断：

假如向这个表中输入一个新记录，何种情况下会违反这条规则呢？

如果不向这个表中输入新记录，会违反这条规则吗？

如果删除这个表中一个记录，会违反这条规则吗？

判定引发违反该规则的操作后，将它们记录下来；留待下一步中使用。在 RDBMS 中实现这个数据库时，这些信息也能帮助你以最为有效的方式建立这条规则。

有一点值得注意：当判定删除一个记录会违反该规则，就必须相应地更改该关系当前的删除规则或为该关系添加一条删除规则。

第 10 章“表关系”中曾提到，无须为删除关系中子表的记录而担忧，因为这一做法不会带来不良影响。现在必须修改这一论断，声明删除子表中的记录会违反必要的业务规则。解决这个问题可以通过为该子表建立一条限制删除规则。**务必确保**当你决定测试业务规则时，记得这一点。

当向 INSTRUCTOR CLASSES 表中插入一个记录时，就会测试该舞蹈室数据库的新业务规则；至多可以将 8 个记录与特定教员联系起来。当删除 INSTRUCTOR CLASSES 表中一个记录时，也会测试该业务规则；每个教员必须至少与一门课程相关联。另外，必须为该规则建立一条限制删除规则。图 11.11 所示为你对相应关系示意图所做的调整。

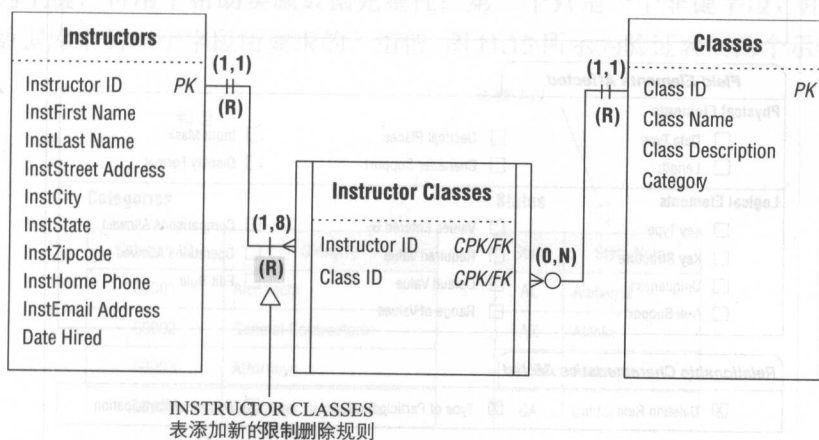


图 11.11 为 INSTRUCTOR CLASSES 表支持新业务规则建立限制删除规则

第 6 步：将该规则记录在业务规则规范表中

最后，填写第 4 步中建立的业务规则的规范表。图 11.12 所示为新规则的完整业务规则规范表。

图 11.12 业务规则规范表

使用验证表支持业务规则

表 11.12

图 11.12 业务规则规范表

BUSINESS RULE SPECIFICATIONS		
Rule Information		
Statement: An instructor must teach one class, but no more than eight (8) classes.		
Constraint: The participation of INSTRUCTORS within the relationship is Mandatory. Also, a single record in INSTRUCTORS can be related to only eight (8) records in INSTRUCTOR CLASSES.		
Type: <input checked="" type="checkbox"/> Database Oriented <input type="checkbox"/> Application Oriented	Category: <input type="checkbox"/> Field Specific <input checked="" type="checkbox"/> Relationship Specific	Test On: <input checked="" type="checkbox"/> Insert <input type="checkbox"/> Update <input checked="" type="checkbox"/> Delete
Structures Affected		
Field Names:		
Table Names: INSTRUCTORS, INSTRUCTOR CLASSES		
Field Elements Affected		
Physical Elements <input type="checkbox"/> Data Type <input type="checkbox"/> Decimal Places <input type="checkbox"/> Input Mask <input type="checkbox"/> Length <input type="checkbox"/> Character Support <input type="checkbox"/> Display Format		
Logical Elements <input type="checkbox"/> Key Type <input type="checkbox"/> Values Entered By <input type="checkbox"/> Comparisons Allowed <input type="checkbox"/> Key Structure <input type="checkbox"/> Required Value <input type="checkbox"/> Operations Allowed <input type="checkbox"/> Uniqueness <input type="checkbox"/> Default Value <input type="checkbox"/> Edit Rule <input type="checkbox"/> Null Support <input type="checkbox"/> Range of Values		
Relationship Characteristics Affected		
<input checked="" type="checkbox"/> Deletion Rule <input checked="" type="checkbox"/> Type of Participation <input checked="" type="checkbox"/> Degree of Participation		
Action Taken		
The type of participation for the INSTRUCTOR CLASSES table was changed to Mandatory. The degree of participation for the INSTRUCTOR CLASSES table was changed to (1,8). A new Restrict deletion rule was added to the relationship for the INSTRUCTOR CLASSES table.		

图 11.12 新规则的完整业务规则规范表

验证表

定义字段特有业务规则时，可能出现这样的情况：一条规则所施加的限制，为给定字段的值的范围定义了一组不同的有效值。（这影响到该字段的字段说明中值的范围元素。）通常，这组值包含的条目数量相对固定，值本

身也很少变化。不过，如果条目的数量相当多，可能发现实施该规则会有些困难。例如，当尝试一一列举字段说明表上值的范围元素中的值时，表中的空白部分可能无法容纳所有的值，在 RDBMS 中实施整组值也会有些麻烦。为避免这一问题，可以将所有的值存储在验证表中。

什么是验证表

第 3 章“术语”中已经提到，验证表（也称为查找表）存储专门用于实现数据完整性的数据。一旦将要求的数据填入表中，就不会经常插入、更新或删除该表中的任何记录。验证表通常（但并非始终）包含两个字段：第一个作为主键，将用于帮助实施数据完整性；第二个只是一个非键字段，用于存储数据库中另一个字段所要求的一组值。图 11.13 所示为验证表的两个示例。

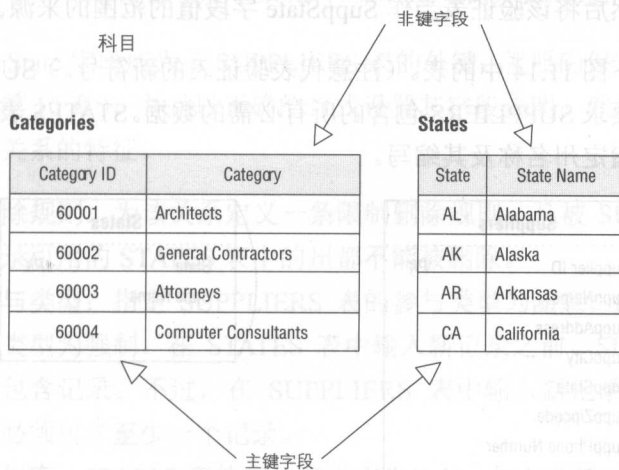


图 11.13 验证表的示例

这一部分介绍的是如何使用主键字段有效实施业务规则。第 12 章“视图”则将介绍如何使用非键字段。

使用验证表支持业务规则

当一条业务规则要限制一个字段的值的范围时，可以使用验证表实施这个限制；然后，该字段就会从验证表中的相应字段提取它的值。建立这种规

则分为两步：定义受该规则影响的字段的父表和验证表之间的关系，和修改该父表中受影响字段的字段说明的值的范围元素。

例如，假设正在处理 SUPPLIERS 表的 SuppState 字段，定义了如下业务规则：

我们选择的任何供货商必须来自美国西部的 11 个州、阿拉斯加或夏威夷。

从中可知，这条规则对 SuppState 字段的值的范围施加了限制，范围限制在 AK、AZ、CA、CO、HI、ID、MT、NM、NV、OR、UT、WA 和 WY（美国 13 个州的字母缩写）。（根据这条规则，不能使用出自其他州的供货商。建立这条规则最简单也最为高效的方法就是，将这些值存储在 STATES 这个验证表中，然后将该验证表当作 SuppState 字段值的范围的来源。

考虑一下图 11.14 中的表。（注意代表验证表的新符号。）SUPPLIERS 表存储该机构要求 SUPPLIERS 包含的所有必需的数据。STATES 表是新的验证表，将存储指定州名称及其缩写。

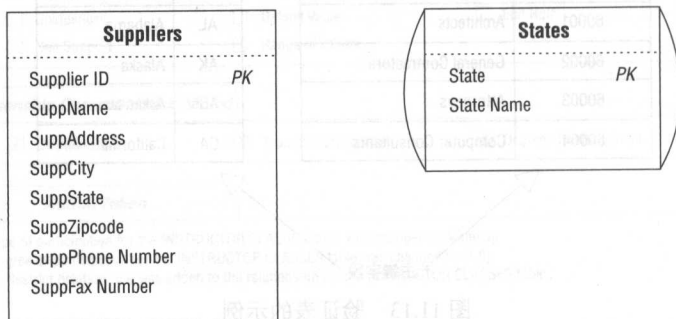


图 11.14 SUPPLIERS 表和 STATES 验证表

首先是建立这两个表之间的关系。从上可知，它们之间存在一对多关系。STATES 表中的单一记录能与 SUPPLIERS 中一个或多个记录相关联，但 SUPPLIERS 表中单一记录仅与 STATES 表中一个记录相关联。前面已经提到，建立一对多关系，就要复制相应父表的主键，将它添加到子表中，作为其外键。尽管 SUPPLIERS 表已经有一个字段 SuppState，但是你将用验证表

STATES 中的 State 字段取代它。(这一修改合情合理,因为它依据的是理想字段的要素,并且与建立一对多关系的方式保持一致。)图 11.15 所示为这两个表的新关系示意图。

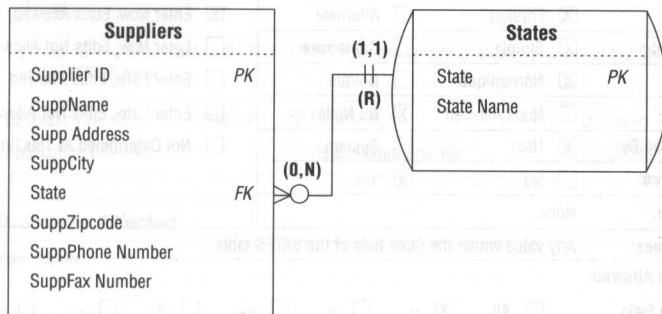


图 11.15 SUPPLIERS 和 STATES 表的关系示意图

既然 State 字段成为了 SUPPLIERS 表的外键,就要确保它满足外键的要素(参见第 10 章),并且以正确的方式设置其字段说明。然后,根据以下要求设置该关系的特征。

- **删除规则:** 为该关系定义一条限制删除规则。凡被 SUPPLIERS 表中记录引用的 STATES 表中的州都不能被删除。
- **参与类型:** 指定 SUPPLIERS 表的参与类型为随意, STATES 表的参与类型为强制。在 STATES 表中输入新记录之前, SUPPLIERS 表无须包含记录。不过,在 SUPPLIERS 表中输入新记录之前, STATES 表必须包含至少一个记录。
- **参与度:** STATES 表的参与度指定为(1,1);如上所述,在 SUPPLIERS 表中输入新记录之前, STATES 表必须包含至少一个记录。SUPPLIERS 表的参与度指定为(0,N);这个表中任意数量的记录都能与表中特定记录相关联。

接下来,将 SUPPLIERS 表中 State 字段说明的值的范围元素修改为如下设置:

STATES 表 State 字段中的任意值。

图 11.16 所示为这一字段的字段说明表中逻辑元素部分所做修改。

Logical Elements							
Key Type:	<input type="checkbox"/> Non	<input type="checkbox"/> Primary	Edit Rule: <input checked="" type="checkbox"/> Enter Now, Edits Allowed <input type="checkbox"/> Enter Now, Edits Not Allowed <input type="checkbox"/> Enter Later, Edits Allowed <input type="checkbox"/> Enter Later, Edits Not Allowed <input type="checkbox"/> Not Determined At This Time				
	<input checked="" type="checkbox"/> Foreign	<input type="checkbox"/> Alternate					
Key Structure:	<input checked="" type="checkbox"/> Simple	<input type="checkbox"/> Composite					
Uniqueness:	<input checked="" type="checkbox"/> Non-unique	<input type="checkbox"/> Unique					
Null Support:	<input type="checkbox"/> Nulls Allowed	<input checked="" type="checkbox"/> No Nulls					
Values Entered By:	<input checked="" type="checkbox"/> User	<input type="checkbox"/> System					
Required Value:	<input type="checkbox"/> No	<input checked="" type="checkbox"/> Yes					
Default Value:	None						
Range of Values:	Any value within the State field of the STATES table						
Comparisons Allowed:							
<input checked="" type="checkbox"/> Same Field	<input type="checkbox"/> All	<input checked="" type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/> ≠	<input type="checkbox"/> <	<input type="checkbox"/> <=
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/> ≠	<input type="checkbox"/> <	<input type="checkbox"/> <=
<input checked="" type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input checked="" type="checkbox"/> =	<input type="checkbox"/> >	<input type="checkbox"/> >=	<input type="checkbox"/> ≠	<input type="checkbox"/> <	<input type="checkbox"/> <=
Operations Allowed:							
<input type="checkbox"/> Same Field	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation	
<input type="checkbox"/> Other Fields	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation	
<input type="checkbox"/> Value Expression	<input type="checkbox"/> All	<input type="checkbox"/> +	<input type="checkbox"/> -	<input type="checkbox"/> x	<input type="checkbox"/> ÷	<input type="checkbox"/> Concatenation	

图 11.16 SUPPLIERS 表中外键字段 State 的逻辑元素部分设置

现在，必须决定采用哪种操作测试这条规则。当使用验证表实施业务规则时，通常是在用户向该字段插入一个新值或更新一个现有值时，测试该规则。另外，无论何种情况，用户输入相应验证表中不存在的值时，都会违反该规则。

最后，为刚建立的业务规则填写业务规则规范表。确保显示出对该字段和新关系所做修改。图 11.17 所示为新规则的完整业务规则规范表。

BUSINESS RULE SPECIFICATIONS		
Rule Information		
Statement: Any supplier we use must be based in one of the 11 contiguous Western states, Alaska, or Hawaii.		
Constraint: Entries for the State field in the SUPPLIERS table are limited to existing values of the State field in the STATES table.		
Type: <input checked="" type="checkbox"/> Database Oriented <input type="checkbox"/> Application Oriented Category: <input checked="" type="checkbox"/> Field Specific <input type="checkbox"/> Relationship Specific Test On: <input checked="" type="checkbox"/> Insert <input checked="" type="checkbox"/> Update <input type="checkbox"/> Delete		
Structures Affected		
Field Names: STATE		
Table Names: SUPPLIERS, STATES		
Field Elements Affected		
Physical Elements <input type="checkbox"/> Data Type <input type="checkbox"/> Decimal Places <input type="checkbox"/> Input Mask <input type="checkbox"/> Length <input type="checkbox"/> Character Support <input type="checkbox"/> Display Format		
Logical Elements <input type="checkbox"/> Key Type <input type="checkbox"/> Values Entered By <input type="checkbox"/> Comparisons Allowed <input type="checkbox"/> Key Structure <input type="checkbox"/> Required Value <input type="checkbox"/> Operations Allowed <input type="checkbox"/> Uniqueness <input type="checkbox"/> Default Value <input type="checkbox"/> Edit Rule <input type="checkbox"/> Null Support <input checked="" type="checkbox"/> Range of Values		
Relationship Characteristics Affected		
<input checked="" type="checkbox"/> Deletion Rule <input checked="" type="checkbox"/> Type of Participation <input checked="" type="checkbox"/> Degree of Participation		
Action Taken		
<p>The Range of Values was set to "Any value within the State field of the STATES table."</p> <p>The type of participation for each table was changed: STATES is Mandatory; SUPPLIERS is Optional.</p> <p>The degree of participation for each table was changed: SUPPLIERS is (0,N); STATES is (1,1).</p> <p>A Restrict deletion rule was defined for the relationship between SUPPLIERS and STATES</p>		

图 11.17 新业务规则的完整规范表

评审业务规则规范表

建立业务规则之后，评审其规范表。认真检查每个规范表，确保规则建立规范，规范表中所有区域填写正确。如发现错误，立即进行必要修改，并再次评审。重复这一过程，直至评审完所有业务规则。

业务规则是数据库的重要组成部分。它们有助于维持整体数据完整性，实施机构特定的完整性限制。从上可知，这些规则根据机构运作和开展的方式，帮助确保数据的有效性和一致性。此外，这些规则最终将影响 RDBMS 中实施该数据库的方式，以及该数据库终端用户应用程序的设计和开发。

必须清楚地知道一点：再访问这些规则将成为常态。例如，当评审最终结构时，也许会发现需要额外的业务规则。也可能发现一些规则并不能提供最初设想的结果，所以需要对他们进行修改。还可能认定一些规则根本就是多余的。（在这种情况下，必须先仔细检查这些规则，然后再决定是否将它们移除。）

记住，当前定义的业务规则以后必定需要修改；极有可能，由于机构运作和开展业务的方式发生改变，需要及时添加业务规则。修改现有业务规则或开发新业务规则都十分正常，因为机构势必会发展壮大、走向成熟，它对外部力量的作用和反应也是如此。这些力量影响机构认知和使用其数据的方式，转而改变机构业务规则的要求。

和数据库设计过程中的许多其他任务一样，定义和建立业务规则的任务仍将继续。如果要为此重复多次，也无须感到沮丧。因为，付出终究会得到丰厚的回报。

案例分析

现在，该为迈克自行车行建立业务规则了。你安排了与迈克和他的员工会谈，借此机会评审数据库中的表和关系。第一项任务是定义和建立字段特有的业务规则。

从评审 PRODUCTS 表入手。检查每个字段的同时，判断它是否需要限制。当检查到 Category 字段时，记起它的值的范围方面存在一个问题。（参见第 9 章“字段说明”中的案例分析。）于是，你再次与迈克和他的员工讨论了这个问题，最终你们就一份不同类别的列表达成共识。然后，迈克决定 Category 字段的值应该限制为这个列表中的项，以确保员工不会随意创造新类别。根据迈克的决定，你定义了一条合适的业务规则，建立了限制：

不允许出现无效产品类别。

在该类别列表中有许多个项，所以你认定最佳的方法是使用验证表。先创建一个新表 CATEGORIES，然后建立它与 PRODUCTS 表之间的关系。接着，画出该关系的示意图，以正确的方式设置该关系的特征。图 11.18 所示为其结果。

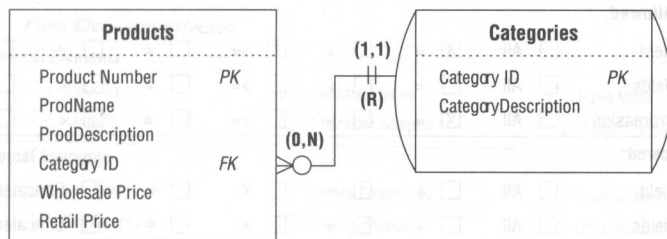


图 11.18 PRODUCTS 和 CATEGORIES 表的关系示意图

下列为该关系特征的设置。

- 该关系需要一条限制删除规则。
- CATEGORIES 表的参与类型为强制。
- PRODUCTS 表的参与类型为随意。
- CATEGORIES 表的参与度为(1,1)。
- PRODUCTS 表的参与度为(0,N)。

记住，通过建立这一关系，将 PRODUCTS 表中原有 Category 字段替换为新 CATEGORIES 表中 Category ID 字段的副本。必须立即确保 PRODUCTS 表中 Category ID 字段满足外键的要素，然后对其字段说明做出合适的改动。最后，修改该字段值的范围元素设置如下：

CATEGORIES 表中 CATEGORY ID 字段的任意值。

图 11.19 所示为 PRODUCTS 表中 Category ID 字段说明的逻辑元素部分修改后的设置。

Logical Elements	
Key Type:	<input type="checkbox"/> Non <input type="checkbox"/> Primary <input checked="" type="checkbox"/> Foreign <input type="checkbox"/> Alternate
Key Structure:	<input checked="" type="checkbox"/> Simple <input type="checkbox"/> Composite
Uniqueness:	<input checked="" type="checkbox"/> Non-unique <input type="checkbox"/> Unique
Null Support:	<input type="checkbox"/> Nulls Allowed <input checked="" type="checkbox"/> No Nulls
Values Entered By:	<input checked="" type="checkbox"/> User <input type="checkbox"/> System
Required Value:	<input type="checkbox"/> No <input checked="" type="checkbox"/> Yes
Default Value:	None
Range of Values:	Any value within the Category ID field in the CATEGORIES table
Comparisons Allowed:	
<input checked="" type="checkbox"/> Same Field <input type="checkbox"/> All <input checked="" type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=	
<input type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=	
<input checked="" type="checkbox"/> Value Expression <input type="checkbox"/> All <input checked="" type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=	
Operations Allowed:	
<input type="checkbox"/> Same Field <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation	
<input type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation	
<input type="checkbox"/> Value Expression <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation	

图 11.19 PRODUCTS 表中 CATEGORY ID 字段说明的逻辑元素部分修改后的设置

现在，必须决定什么时候应该测试该规则。从上文中可知，当用户向该字段中插入一个值或更新其中的已有值时，通常可以测试借助验证表建立的规则。

最后，完成这条新业务规则的规范表。这个规范表反映的是对 Category ID 字段说明以及 CATEGORIES 和 PRODUCTS 表间关系的特征所做的改动。图 11.20 所示为完善的业务规则规范表。

对这个表中剩余的字段以及其他表的所有字段逐一重复这个过程。完成后，继续下一任务。

BUSINESS RULE SPECIFICATIONS		
Rule Information		
Statement: Invalid product categories are not allowed.		
Constraint: Entries for the Category ID field in the PRODUCTS table are limited to existing values of the Category ID field in the CATEGORIES table.		
Type: <input checked="" type="checkbox"/> Database Oriented <input type="checkbox"/> Application Oriented	Category: <input checked="" type="checkbox"/> Field Specific <input type="checkbox"/> Relationship Specific	Test On: <input checked="" type="checkbox"/> Insert <input checked="" type="checkbox"/> Update <input type="checkbox"/> Delete
Structures Affected		
Field Names: CATEGORY ID		
Table Names: PRODUCTS, CATEGORIES		
Field Elements Affected		
Physical Elements		
<input type="checkbox"/> Data Type	<input type="checkbox"/> Decimal Places	<input type="checkbox"/> Input Mask
<input type="checkbox"/> Length	<input type="checkbox"/> Character Support	<input type="checkbox"/> Display Format
Logical Elements		
<input type="checkbox"/> Key Type	<input type="checkbox"/> Values Entered By	<input type="checkbox"/> Comparisons Allowed
<input type="checkbox"/> Key Structure	<input type="checkbox"/> Required Value	<input type="checkbox"/> Operations Allowed
<input type="checkbox"/> Uniqueness	<input type="checkbox"/> Default Value	<input type="checkbox"/> Edit Rule
<input type="checkbox"/> Null Support	<input checked="" type="checkbox"/> Range of Values	
Relationship Characteristics Affected		
<input checked="" type="checkbox"/> Deletion Rule	<input checked="" type="checkbox"/> Type of Participation	<input checked="" type="checkbox"/> Degree of Participation
Action Taken		
The Range of Values was set to "Any value within the CategoryID field of the CATEGORIES table." The type of participation for each table was changed: PRODUCTS is Optional; CATEGORIES is Mandatory. The degree of participation for each table was changed: PRODUCTS is (0,N); CATEGORIES is (1,1). A Restrict deletion rule was defined for the relationship between PRODUCTS and CATEGORIES.		

图 11.20 新业务规则完善后的规范表

下一任务是建立关系特有业务规则。首先，评审 EMPLOYEES 和 INVOICES 表之间的关系，再评审其关系示意图，判断该关系是否需要限制。

一切似乎都正常，所以继续评审 VENDORS 和 PRODUCTS 表之间的关系。图 11.21 所示为这两表的关系。

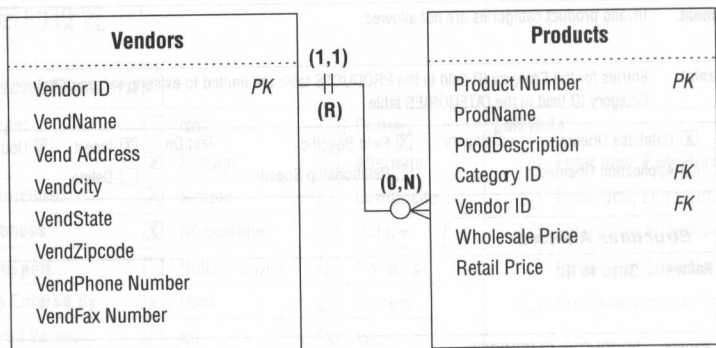


图 11.21 VENDORS 和 PRODUCTS 表的关系示意图

就在你和迈克讨论是否需要对这个关系施加限制时，迈克决定 PRODUCTS 表应该有一个限制。他想要确保 VENDORS 表中每个供应商都至少与一种产品相关联；他认为无须保留不供应任何产品的供应商。所以，你为这个限制定义了如下业务规则：

每个供应商必须至少供应一种产品。

通过修改相应关系特征，你建立了这条规则。先将表的参与类型和参与度分别指定为强制和(1,N)。然后，基于 PRODUCTS 表，为该关系定义了一条限制删除规则；这能有效避免不慎删除与给定供应商相关联的唯一产品。图 11.22 所示为修改的结果。

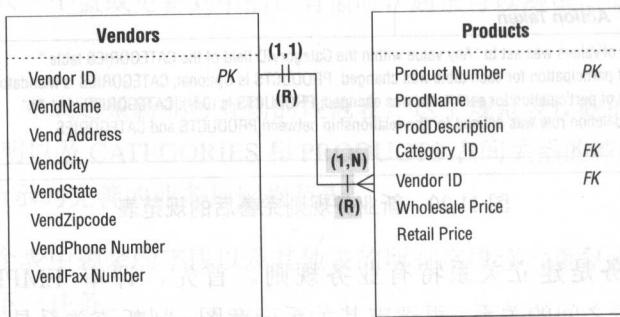


图 11.22 VENDORS 和 PRODUCTS 表修改后的关系示意图

当用户向 PRODUCTS 表插入一个记录或删除其中一个记录时，将会测试这种业务规则。所以填写这条规则的业务规则规范表，从而完成这一过程。图 11.23 所示为完成后的规范表。

BUSINESS RULE SPECIFICATIONS		
Rule Information		
Statement: Every vendor must supply at least one product.		
Constraint: A single record in the VENDORS table must be associated with at least one record in the PRODUCTS table.		
Type: <input checked="" type="checkbox"/> Database Oriented <input type="checkbox"/> Application Oriented	Category: <input checked="" type="checkbox"/> Field Specific <input type="checkbox"/> Relationship Specific	Test On: <input checked="" type="checkbox"/> Insert <input type="checkbox"/> Update <input checked="" type="checkbox"/> Delete
Structures Affected		
Field Names:		
Table Names: VENDORS, PRODUCTS		
Field Elements Affected		
Physical Elements		
<input type="checkbox"/> Data Type	<input type="checkbox"/> Decimal Places	<input type="checkbox"/> Input Mask
<input type="checkbox"/> Length	<input type="checkbox"/> Character Support	<input type="checkbox"/> Display Format
Logical Elements		
<input type="checkbox"/> Key Type	<input type="checkbox"/> Values Entered By	<input type="checkbox"/> Comparisons Allowed
<input type="checkbox"/> Key Structure	<input type="checkbox"/> Required Value	<input type="checkbox"/> Operations Allowed
<input type="checkbox"/> Uniqueness	<input type="checkbox"/> Default Value	<input type="checkbox"/> Edit Rule
<input type="checkbox"/> Null Support	<input type="checkbox"/> Range of Values	
Relationship Characteristics Affected		
<input checked="" type="checkbox"/> Deletion Rule	<input checked="" type="checkbox"/> Type of Participation	<input checked="" type="checkbox"/> Degree of Participation
Action Taken		
The type of participation for PRODUCTS was changed to Mandatory. The degree of participation for PRODUCTS was changed to (1,N). A Restrict deletion rule was defined for the PRODUCTS table.		

图 11.23 完成后的业务规则规范表

对剩余所有关系重复这一过程。完成后,就可以进入数据库设计的下一阶段了。

小结

本章以为**业务规则**下定义开篇,介绍了业务规则是根据机构认知和使用其数据的方式,对一个字段或关系所施加的限制。而机构认知和使用其数据的方式则是源于机构运作和开展业务的方式。业务规则主要分为两种:**面向数据库**和**面向应用程序**。尽管本书的重点是面向数据库业务规则,但是至少可以记录面向应用程序业务规则的基本元素,留待后续实现过程使用。

面向数据库业务规则也分为两类:**字段特有业务规则**,影响的是特定字段的字段说明的元素;**关系特有业务规则**,影响的是一个关系的特征。

接着,本章讨论了定义和建立业务规则。这涉及与用户和管理人员合作,定义该机构所要求的业务规则。建立业务规则应分先后顺序,先建立**字段特有业务规则**,再建立**关系特有业务规则**。

然后,介绍了定义和建立每种业务规则的必要步骤。一般而言,读者需要处理的是一个字段或关系;根据该规则评审该字段或关系,判断是否需要限制;定义合适的业务规则;修改相应的字段说明元素或关系特征,建立该规则;决定应采取的测试操作;最后,为该规则完成业务规则规范表。

随后,本章又介绍了业务规则规范表的元素以及如何定义规范表中的规则。使用业务规则规范表能将所有规则记录在文档中,并提供记录和评审这些规则的标准方法。

本章结尾讨论了**验证表**。对于限制特定字段值的范围的业务规则,创建和使用验证表能对其提供支持。以这种方式,验证表能帮助实施完整性。使用验证表需要建立新关系,这些关系和数据库中其他类型的关系一样,拥有相同的特征。

思考题

1. 什么是业务规则？
2. 说出两种主要业务规则的名称。
3. 在数据库逻辑设计过程中，能建立面向应用程序业务规则吗？
4. 面向数据库业务规则分为哪两类？
5. 什么是字段特有业务规则？
6. 何种情况下测试业务规则？
7. 如何将业务规则记录在文档中？
8. 说出业务规则规范表的两个优点。
9. 业务规则规范表中采取的措施部分用途是什么？
10. 验证表的用途是什么？
11. 验证表的典型结构是什么？
12. 业务规则和验证表之间有什么联系？
13. 为什么要评审所有完成的业务规则规范表？

利用所有关系表完成。完成以后，还可以进入数据库设计的下一
阶段了。

第 12 章

视图

从宇宙的角度看地球，真可谓一览无余。

——费奥多尔·米哈洛维奇·陀思妥耶夫斯基

本章内容

什么是视图

视图之剖析

确立视图

案例分析

小结

思考题

什么是视图

第3章“术语”中已经提到，视图是由数据库中一个或多个表中字段组成的虚拟表；它也可以包含取自其他视图中的字段。组成一个视图的表和视图被称为该视图的基表。视图是“虚拟的”，因为它只是从基表中提取数据，而本身并不存储数据。实际上，视图被存储在数据库中的唯一信息是它的结构；每次以某种方式访问视图，RDBMS 都会重建和“重新植入”该视图。

许多主流 RDBMS 程序都支持视图，但是一些将视图称为已保存的查询。所以其名称由具体使用的 RDBMS 程序决定。

❖注意：尽管所有主流数据库供应商都支持本文所描述的视图，但是有一些供应商支持所谓的索引（或物化）视图。索引视图与一般视图的区别在于它存储数据，索引其字段能提升相应 RDBMS 处理视图数据的速度。本书不会详细讨论索引视图，因为它是特定供应商的实现问题。不过，如果读者正在处理客户/服务器或大型机 RDBMS 程序，可以自行深入研究这个问题。

视图能让你从多个不同的方面查看数据库中的信息，也能在处理数据时提供极大的灵活性。创建视图的方式多种多样，在基于多个相关表的情况下尤其管用。

以下为数据库中定义和使用视图的几个理由。

- 可同时处理取自多个表中的数据。在数据库设计过程中，已建立了各个表之间的一对多关系或多对多关系。（前面提到，借助联系表建立多对多关系。）视图提供的机制，能让你同时处理取自多个相关表中的数据。
- 反映最新的信息。因为每次访问视图，RDBMS 都会重建和重新植入它，视图所展示的信息显示了其基表中数据的最新变化。
- 根据个人或群体的特定需求定制。建立视图适应各种要求，比如为某份报表提供数据或检查机构内多个部门的某些通用信息。
- 有助于实施数据完整性。以定义验证表的方式定义验证视图，其用途是为数据库中的给定字段提供有效的值的范围。
- 用于保密和安全用途。从其基表中选择字段定义视图，判断特定用户或用户群体可访问的数据。

定义视图应做到态度认真、操作熟练。等到在 RDBMS 中实现数据库时，视图就会是一笔可观的财富。

视图之剖析

设计数据库逻辑结构时，可以定义的视图分为三种：数据视图、聚合视图和验证视图。在 RDBMS 中实现数据库时，可以定义的视图则包括物化视图和分割视图两种。定义后两种视图的能力及其方式主要由使用的 RDBMS 决定，所以不在本书讨论范围内。本书将着重讨论前面三种视图。

数据视图

这种视图用于检查和操作来自一个或多个基表的数据。

单表数据视图

尽管能使用所有该基表中的字段建立这种视图，但通常使用的只是选定字段。（使用某基表中所有字段建立视图，所产生的只是该基表的虚拟副本。）例如，如果想要制作机构内部人人都可以使用的通讯表，其中包含员工姓名和电话号码。基于 EMPLOYEES 表，选取其中的 Employee ID、EmpFirst Name、EmpLast Name 和 EmpPhone Number 字段，创建一个视图 EMPLOYEE PHONE LIST。图 12.1 所示为这个视图的示意图。（注意表示视图的新符号。）

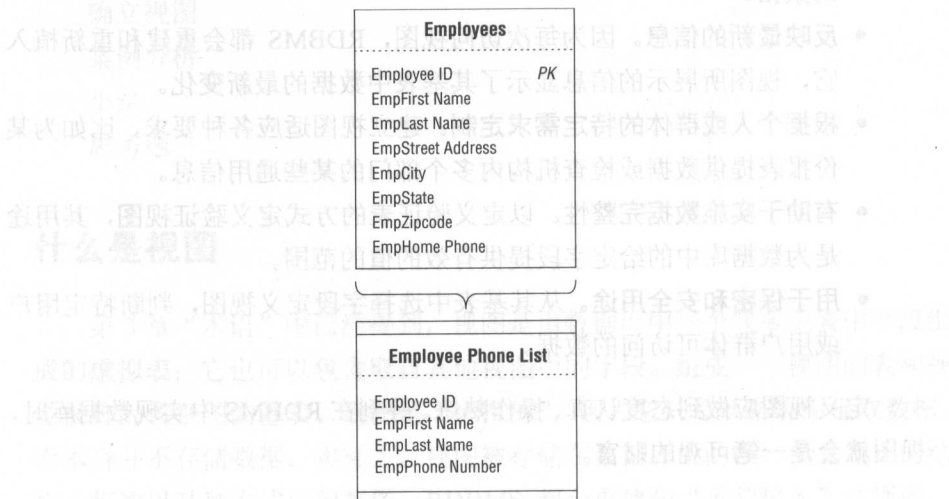


图 12.1 EMPLOYEE PHONE LIST 视图

每次访问 EMPLOYEE PHONE LIST 视图，RDBMS 都会重建和重新植入它。这个视图将反映出 EMPLOYEES 表中数据的最新变化。图 12.2 所示为 RDBMS 通常展示视图中数据的方式。注意，视图的外观与表的十分相似；这也是视图被称为“虚拟表”的另一原因。

Employee Phone List

Employee ID	EmpFirst Name	EmpLast Name	EmpPhone Number
100	Zachary	Erlch	553-3992
101	Susan	Black	790-3992
102	Joe	Rosales	551-4993
103	Alastair	Black	227-4992
104	Katie	Christian	525-2993
105	Diana	Price	248-4953

图 12.2 EMPLOYEE PHONE LIST 视图中的信息

你可以随时修改单表数据视图中的数据，修改会透过视图，反映到基表中。不过，字段说明和业务规则都会限定对数据所做修改的类型。例如，EmpLast Name 字段说明的 null 支持元素设置为“禁用 null”，你就不能删除视图中的姓氏。

❖ 注意：在大多数 RDBMS 软件中，视图实现过程存在一定程度的差别。必须检查所用 RDBMS 的说明材料，确认该 RDBMS 支持视图的充分程度及其对修改视图中数据所施加限制的类型。

多表数据视图

本章开始已经提到，可以使用多个表定义一个数据视图。唯一的要求是创建视图所用的表必须彼此之间存在关系；这能有效确保视图所呈现的信息有效且具有意义。例如，假设你正在为当地的社区大学设计数据库，图 12.3 所示的表是数据库的一部分。你恰巧认为需要创建一个视图 CLASS ROSTER，显示每门课程的名称和当前选修课程的学生姓名。这个任务比较简单，可以使用这三个表作为视图的基表；它们包含了定义视图所需的字段，并且彼此之间具有关系。

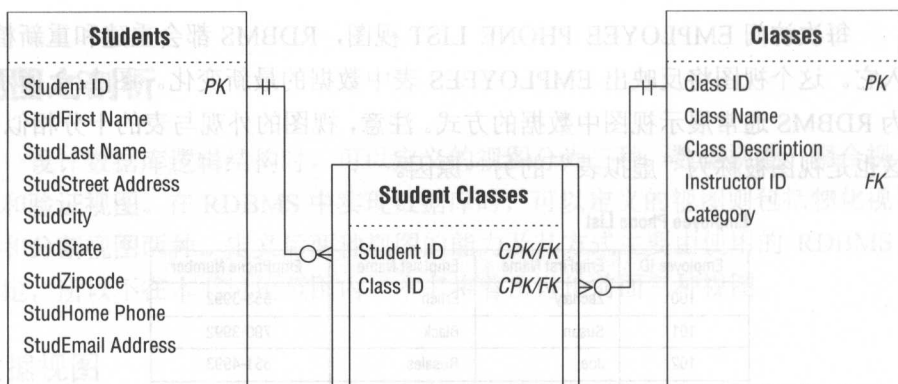


图 12.3 CLASS ROSTER 视图的基表

于是，使用 CLASSES 表中的 Class Name 字段和 STUDENTS 表中的 StudFirst Name 和 StudLast Name 字段，定义 CLASS ROSTER 视图。每门课程都会出现相应的学生姓名，CLASSES 和 STUDENTS 通过联系表 STUDENT CLASSES 保持关联。图 12.4 所示为 CLASS ROSTER 视图的示意图。注意，此前所有基表都没有变动。

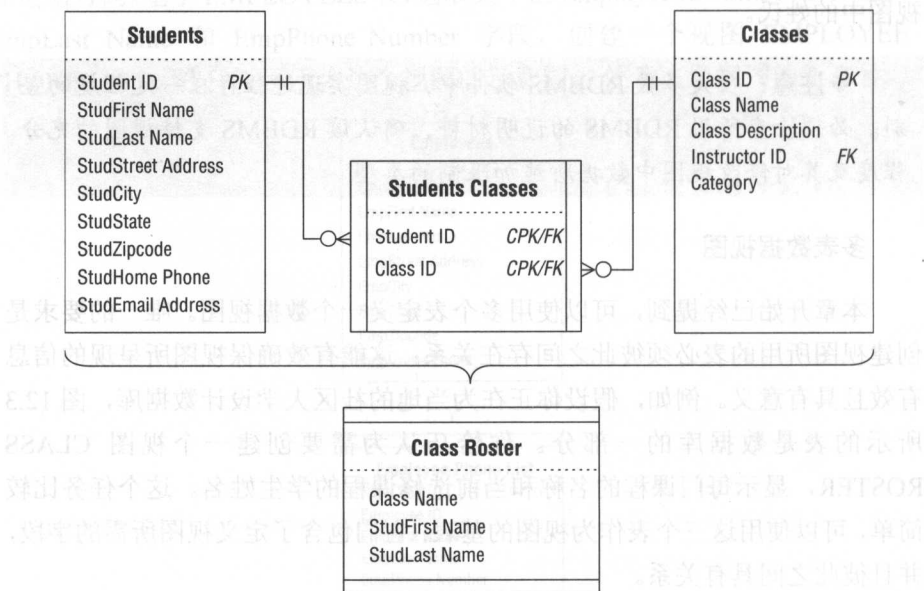


图 12.4 CLASS ROSTER 视图的示意图

每次访问 CLASS ROSTER 视图, RDBMS 都会使用其当前基表中的大部分数据重建和重新植入这个视图。图 12.5 所示为这个视图的数据样本。

Class Roster

Class Name	StudFirst Name	StudLast Name
Advanced Calculus	Martin	Applebee
Advanced Calculus	Gina	Carter
Advanced Calculus	Joe	Rosales
Advanced Calculus	Sara	Ulrich
Advanced Music Theory	Mike	Hernandez
Advanced Music Theory	Susan	Black
Advanced Music Theory	Lee	Turner
American History	Gina	Carter
American History	Susan	Black
American History	George	Price
American History	Joe	Rosales

图 12.5 CLASS ROSTER 视图部分数据示例

随时可以修改多表数据视图中的大多数数据, 修改会通过视图, 反映到基表中。显然, 不能修改基表中的任何主键。和单表视图一样, 字段说明和业务规则决定了对该数据所做修改的类型。(再次提醒, 必须检查所用的 RDBMS 的说明材料, 确认对视图所施加的限制。)

CLASS ROSTER 视图中的冗余数据, 是 CLASSES 表中一个记录和 STUDENTS 表中多个记录合并的结果; 特定课程名称出现的次数与选修该课程的学生人数相等。这种明显的冗余数据是可接受的, 因为该数据并非实际存储在视图中, 而是从视图的基表中提取出来的。在这些基表中, 这些数据的存储符合相应的数据库设计规则。RDBMS 一般以这样的方式展示多个视图中的数据。

还有一点值得注意, 就是数据视图并不具有其自身主键。它缺少主键是因为它不是表; 真正的表存储数据, 需要一个主键作为识别其记录的唯一方式。不过, 如果你认为基表的主键有利于视图提供的信息, 就可以添加该视图的任意基表的主键。

❖ 注意：为了避免不必要的混淆或歧义，制作数据视图的示意图时，必须确保该视图符号内无任何主键图标。

聚合视图

这种视图用于展示特定一组数据聚合所产生的信息。正如数据视图一样，可以使用一个或多个基表定义聚合视图。然后，再加入一个或多个计算字段，由计算字段包含的函数把数据和一个或多个数据字段（取自视图的基表）聚合成聚合数据。总和、平均值（算术平均值）、最小值、最大值和计数都是最常见的聚合函数，可对任意一组数据使用。所有主流 RDBMS 也都支持这些函数。

不妨假设使用图 12.3 所示学校示例中的表，想知道每门课程分别有多少学生选修。你的第一想法是定义数据视图 CLASS REGISTRATION，提供解答所需的信息。因此，使用 CLASSES 表中的 Class Name 字段和 STUDENT CLASSES 表中的 Student ID 建立视图。图 12.6 所示为新 CLASS REGISTRATION 视图的示意图。

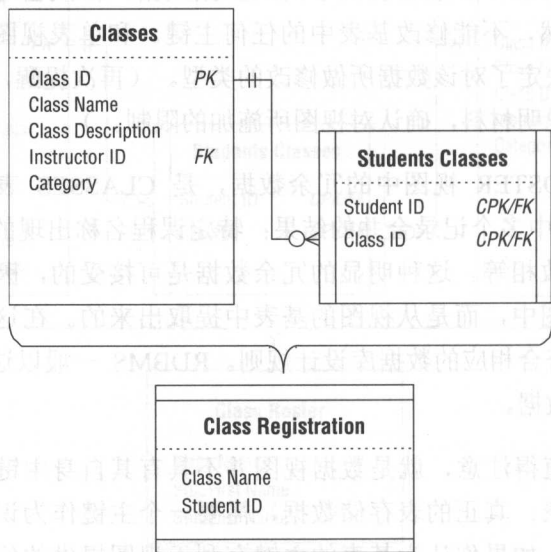


图 12.6 新 CLASS REGISTRATION 视图的示意图

为寻求解答访问这个视图。图 12.7 所示为该视图中数据的部分样本。

Class Registration

Class Name	Student ID
Advanced Calculus	1003
Advanced Calculus	1025
Advanced Calculus	1073
Advanced Calculus	1110
Advanced Music Theory	1045
Advanced Music Theory	1066
Advanced Music Theory	1085
Business Administration	1025
Business Administration	1066
Business Administration	1017
Business Administration	1073

图 12.7 CLASS REGISTRATION 视图中数据的部分样本

现在，必须数清给定课程名称的所有实例，确定选修该课程的学生人数。想想将面临的工作，这可不简单啊！与其干这样乏味的事，倒不如使用聚合视图轻易就能解决（也更为高效）。

不过，无须定义一个新的视图，可以直接在现有的视图上进行修改。使用计算字段 **Total Students Registered** 取代原视图中的 **Student ID** 字段，前者可以计算每门课程的学生数量。（使用计算字段，必须确保其名称意义明确，能与视图中其他字段区别开来。）这个计算字段将使用一个计数函数，计算 **STUDENT CLASSES** 表中与每个 **Class ID** 相关的 **Student ID** 数量。（下文将分别介绍用文档记录视图以及记录计算字段所用表达式的方法。）图 12.8 所示为 **CLASS REGISTRATION** 视图修改后的示意图。

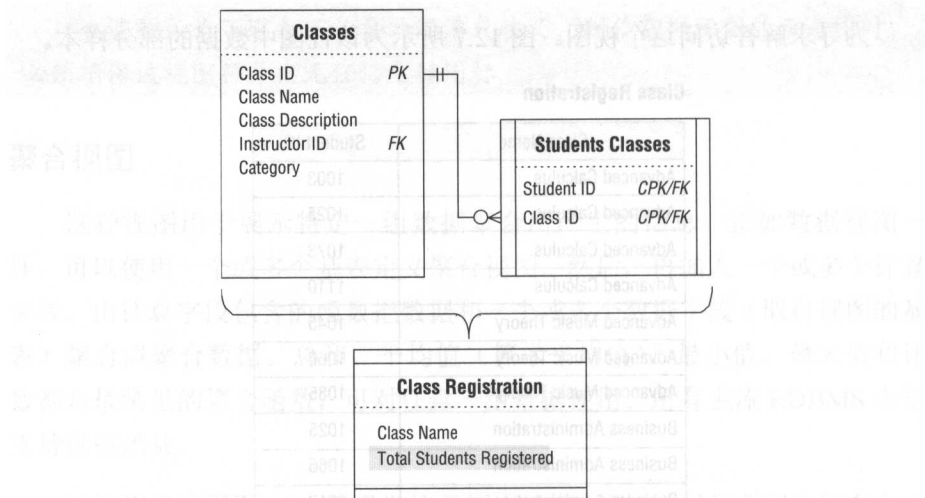


图 12.8 CLASS REGISTRATION 视图修改后的示意图

和数据视图一样，每次访问 CLASS REGISTRATION 视图，RDBMS 就会使用其基表中最新的数据，重建和重新植入它。图 12.9 所示为该视图的数据样本。

Class Registration

Class Name	Total Students Registered
Advanced Calculus	92
Advanced Music Theory	80
Business Administration	84
Computers in Business	98
English Literature	80
Introduction to Biology	60
Introduction to Database Design	84
Pan-American Studies	80

图 12.9 修改后的 CLASS REGISTRATION 视图的数据样本

对于这个视图有以下三点需要注意。

1. Total Students Registered 字段对应每门课程名称分别展示了一个数字，表示选修该课程的总人数。

2. Class Name 字段中的冗余已经消除, 给定课程的所有例子组合成一个例子。因此, Class Name 现在是一个组合字段, 它的值绝不能被修改。

❖ 注意: 聚合视图中所有数据字段都是组合字段。

3. 因为聚合视图完全是由组合字段和计算字段组成的, 所以无法修改其任何数据。

聚合视图的最佳用途是作为报表的基础或提供各种统计信息。下文将介绍如何运用过滤条件, 控制和限制视图展示的数据。

验证视图

验证视图与验证表类似, 它也可以帮助实现数据完整性。当业务规则限制特定字段的值的范围时, 也可以借助验证视图实施该限制, 和使用验证表一样简单。两者的不同之处在于它们的构造: 验证表存储其自身的数据, 而验证视图从其基表中提取数据。另外, 定义验证表可以使用一个或多个基表, 而验证表通常使用一个表, 仅包含两个或三个基表的字段。(验证视图的结构与验证表的结构也极其相似。)

例如, 假设正在为一位小承包商设计数据库, 使用的是图 12.10 中的表。

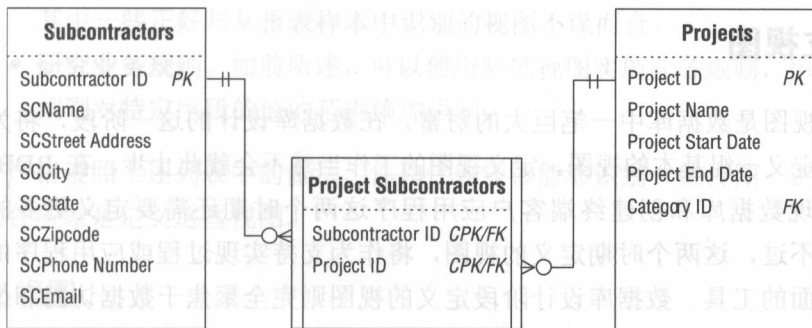


图 12.10 某小承包商数据库中的表

从图中可知, SUBCONTRACTORS 表中 Subcontractor ID 字段为

PROJECT SUBCONTRACTORS 表中 Subcontractor ID 字段提供值的范围。(前面已经提到, 外键的值取自被引用的主键。)不过, 你决定限制当前用户对 SUBCONTRACTORS 表中某些字段的访问, 用户可访问的字段应只包括 Subcontractor ID、SCName、SCPhone Number 和 SCEmail。于是, 定义验证视图 APPROVED SUBCONTRACTORS, 这个验证视图包含这些字段并且仍然提供 PROJECT SUBCONTRACTORS 表中 Subcontractor ID 字段的值的范围。图 12.11 所示为修改后的示意图, 其中包括新的视图。

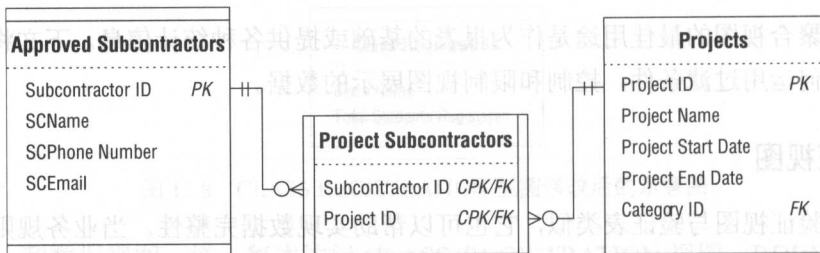


图 12.11 修改后的表示意图和新的 APPROVED SUBCONTRACTORS 视图

这样, APPROVED SUBCONTRACTORS 视图只允许用户访问上述几个字段, 也为 PROJECT SUBCONTRACTORS 表中 Subcontractor ID 字段提供了合适的值的范围。另外, 这个视图将实施 SUBCONTRACTORS 表存在的关系特征, 因为它是视图的基表。

确立视图

视图是数据库中一笔巨大的财富。在数据库设计的这一阶段, 将为数据库定义一组基本的视图。定义视图的工作当然不会就此止步, 在 RDBMS 中实现数据库和创建终端客户应用程序这两个时期还需要定义更多的视图。不过, 这两个时期定义的视图, 将作为支持实现过程或应用程序的特定方面的工具。数据库设计阶段定义的视图则完全聚焦于数据访问和信息检索。

与用户和管理人员合作

你将再次与用户和管理人员的代表合作，识别机构要求的视图类型。识别完后，将建立这些视图并将它们用文档记录，然后你和两方代表评审这些视图，确保定义规范合理。

与代表们开展第一次会谈前，应先回顾整个数据库设计过程所做的记录。目标是了解机构可能需要的视图类型。几乎所有的机构都会投入大量时间制作和阅读报表，所以应该重点关注记录中的这一方面。另外，还要回顾分析阶段收集的报表样本。

当你和代表们会谈时，遵循以下几点将有助于确认视图要求。

- 和代表们一道回顾你的记录。在多数情况下，讨论特定主题将会使人受到启发，发现新视图或要求的视图。例如，围绕任务目标展开讨论时，有人可能会想到需要一个新的视图。
- 回顾设计早期阶段收集的数据输入、报表和演示样本。检查这些样本，尤其是总结式的报表，易于揭示对于某些类型视图的需要。
- 检查表及其所表示的主题。一些代表也许能基于单个主题发现视图需求。当有人提到员工这样的主题时，其他人可能会说，“出于保密考虑，我们确实需要一个视图，用于限制特定的员工数据。”
- 分析表关系。极有可能会发现需要为许多关系创建相应的多表视图。其中一些正好与从报表样本中识别的视图不谋而合。
- 研究业务规则。如前所述，可以使用验证视图实施业务规则，由业务规则对特定字段的值的范围施加限制。

严格遵照上述列表中的做法，你和代表应该能够识别一些视图。识别之后的任务就是定义这些视图。

定义视图

现在，将使用合适的表和字段，定义刚才识别的所有视图。评审关系示意图，识别视图结构所需的表和字段。确定所需的表和字段后，就可以定义视图，并画出其视图示意图。

例如,假设认定可以对图 12.12 所示的报表使用一个视图;新视图的名称为 CUSTOMER CALL LIST。

在设计过程中所做的记录将再次发挥作用。在设计过程的分析阶段,已经评审了这个报表,并且注意到这个报表表示顾客及其订单的信息;根据其订单数据,可以知道给定顾客最后一次采购的时间。现在,评审 CUSTOMERS 和 ORDERS 表的关系示意图,将使用这两个表中的字段创建 CUSTOMER CALL LIST 视图。图 12.13 所示为这两表的关系示意图。

检查完关系示意图,你确定建立这个视图需要使用 5 个字段: CUSTOMERS 表中的 CustFirst Name、CustLast Name、CustPhone Number 和 CustCity,以及 ORDERS 表中的 Order Date。将这些字段分配到该视图中,再记录在视图示意图中, CUSTOMER CALL LIST 视图就定义完成了。完成后的示意图应类似于图 12.14 中的示意图。

Customer Call List			
City	Customer Name	Phone Number	Last Purchase
Bothell	Sara Anderson	542-0039	05/16/02
	Jim Booth	367-4495	02/11/02
	Larry Currey	445-3394	02/06/02
Bellevue	Jim Davis	545-9932	05/10/02
	Larry Lang	545-3384	01/22/02
	Sandra Wasser	367-2293	06/30/02
Edmonds	Julia Black	223-9943	04/12/02
Lynnwood	Mary Black	562-1274	02/28/02
	Barbara Reeves	445-2094	03/07/02

图 12.12 报表需要的视图

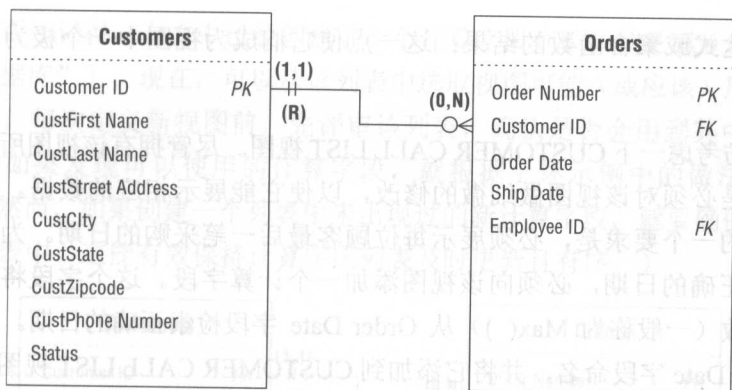


图 12.13 CUSTOMERS 和 ORDERS 表的关系示意图

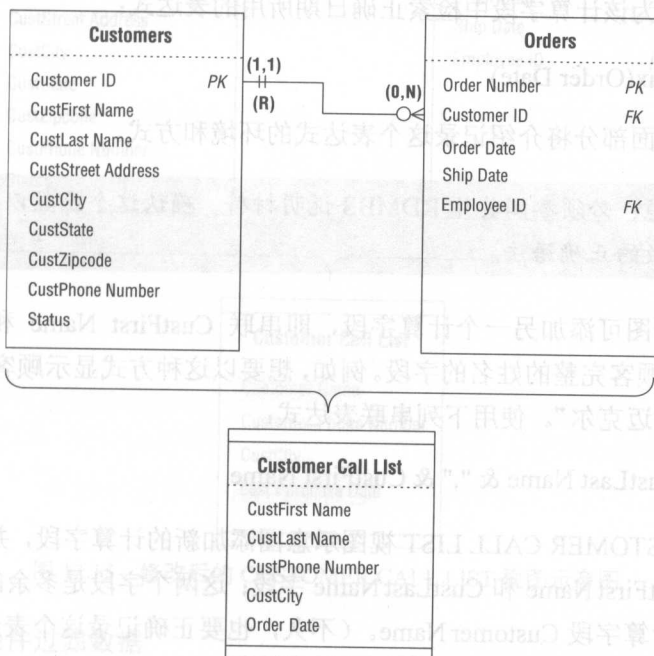


图 12.14 CUSTOMER CALL LIST 视图示意图

酌情使用计算字段

在数据库设计早期，已经知道出于各种原因，表中不能包含计算字段。但是，视图的重要特征之一是能够包含计算字段。计算字段能展示串

联、表达式或聚合函数的结果；这一点使它们成为视图中一个极为灵活的结构。

不妨考虑一下 CUSTOMER CALL LIST 视图。尽管拥有该视图所需的字段，但是必须对该视图做细微的修改，以使它能展示相应的数据。对于这个视图的一个要求是，必须展示每位顾客最后一笔采购的日期。为了检索和展示正确的日期，必须向该视图添加一个计算字段。这个字段将使用最大值函数（一般称为 `Max()`）从 `Order Date` 字段检索正确的日期。为 `Last Purchase Date` 字段命名，并将它添加到 CUSTOMER CALL LIST 视图示意图中。（这个视图不再需要 `Order Date` 字段，所以可以将它从视图结构中移除。）下列为该计算字段中检索正确日期所用的表达式：

`Max(Order Date)`

本节后面部分将介绍记录这个表达式的环境和方式。

❖注意：必须参阅具体 RDMBS 说明材料，确认这个函数以及本章所示其他函数的正确语法。

这个视图可添加另一个计算字段，即串联 `CustFirst Name` 和 `CustLast Name` 显示顾客完整的姓名的字段。例如，想要以这种方式显示顾客姓名：“赫尔南德斯，迈克尔”。使用下列串联表达式：

`CustLast Name & ", " & CustFirst Name`

向 CUSTOMER CALL LIST 视图示意图添加新的计算字段，并从该视图中清除 `CustFirst Name` 和 `CustLast Name` 字段；这两个字段是多余的，因为现在使用了计算字段 `Customer Name`。（不久，也要正确记录这个表达式。）

图 12.15 所示为完成修改后的视图示意图。

从上文可知，计算字段具有很高的价值，可以提升视图所提供信息的水平。本章早前也曾提到，计算字段对于聚合视图十分关键。大量经验表明，当认为可能需要计算字段，且它们能提供相关有价值的信息或能提升相应视图利用数据的方式时，就可以使用计算字段。

是否还记得，上文中已经创建了一个计算字段列表（参见第 6 章“分析现有数据库”）。现在，可以从该列表选取视图可能（或应该）用到的计算字段。每次定义新视图前，先评审该列表，确认是否会用到其中的计算字段。如果发现可以使用的计算字段，就根据上述示例中的做法进行创建。（然而，如果创建一个列表中未出现过的新计算字段，就要确保将它添加到列表中。这能有效保持计算字段列表及时更新且有序。）

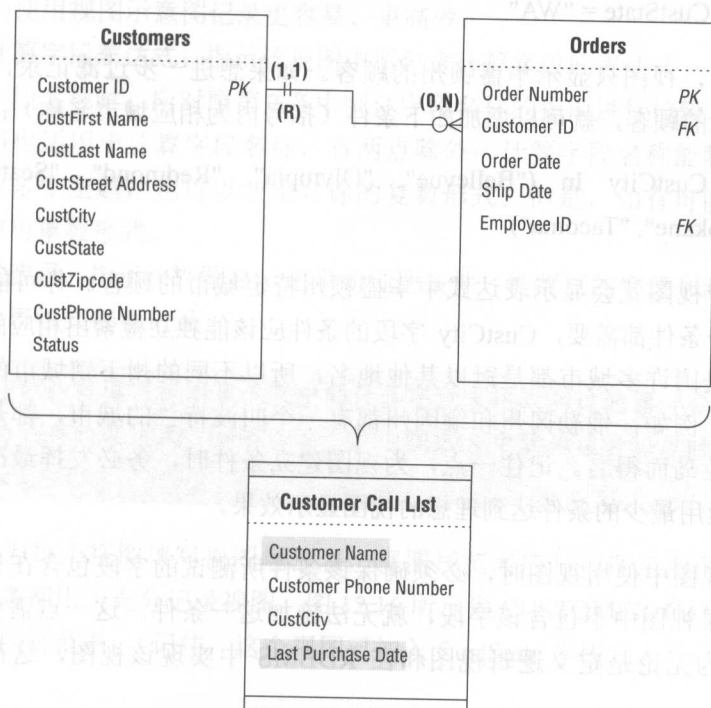


图 12.15 修改后的 CUSTOMER CALL LIST 视图示意图

运用条件过滤数据

视图的另一个极其有用的特征是：对视图中的一个或多个字段施加条件，过滤其展示的记录。例如，CUSTOMER CALL LIST 视图新加入了 CustState 字段。该视图不仅会继续展示先前的记录集合，而且也可以看到每位顾客所在的州。不过，你可能会希望该视图显示特定一组记录，比如家在华盛顿州的顾客。为了实现这个目的，可以对 CustState 字段设置一个

特殊的条件, 过滤部分数据, 这样该视图就只会显示华盛顿州顾客的记录了。

❖注意: 在数据库的工作中, 条件是指测试特定字段的值的表达式。如果该字段的值符合条件, 视图就会包含相应的记录。

下列为过滤 CUSTOMER CALL LIST 视图中记录所用表达式:

```
CustState = "WA"
```

现在, 视图只显示华盛顿州的顾客。如果想进一步过滤记录, 只显示特定城市的顾客, 就可以添加如下条件 (括号内为相应城市名称):

```
CustCity In ("Bellevue", "Olympia", "Redmond", "Seattle",  
"Spokane", "Tacoma")
```

这个视图就会显示表达式中华盛顿州特定城市的顾客。你可能怀疑为什么两个条件都需要, CustCity 字段的条件应该能独立检索出相应的记录。问题是美国许多城市都是冠以其他地名, 所以不同的州下辖城市的名称可能相同。例如, 俄勒冈州和缅因州都有一个叫波特兰的城市, 都是因英国的波特兰岛而得名。记住一点, 为视图建立条件时, 务必发挥最准确的判断——运用最少的条件达到理想的视图显示效果。

在视图中使用视图时, 必须确保该条件所测试的字段包含在视图结构中。如果视图中不包含该字段, 就无法施加这一条件。这一点需要特别注意, 因为无论是定义逻辑视图和在 RDBMS 中实现该视图, 这都是硬性要求。

对视图运用过滤有一个问题, 就是无法在视图示意图中显示; 因此, 必须将它记录在视图规范表中。

使用视图规范表记录视图

每创建一个视图必须对应一个视图规范表。视图规范表用于记录相应视图的特征, 它包含以下几个项:

- 名称: 指示该视图的名称。不过, 记录名称之前, 先对照第7章“建

立表结构”中制订表名称的指南进行检验。这些指南也适用于视图的命名，唯一的区别就是：视图的名称能隐含或显示多个主题。这是因为定义视图可以使用多个基表，所以它们确实表示多个主题。

- **类型**：指示所定义的视图类型为数据、聚合或验证视图。
- **基表**：指示该视图基表的名称。视图示意图也显示了这些基表，在这里显示是为了方便起见。不过，视图规范表并不包含字段名称，因为使用视图示意图记录更容易、更高效。
- **计算字段表达式**：指示该视图中所包含计算字段的表达式。记录计算字段名称时，应对照第7章中制订字段名称的指南进行检验。这些指南也适用于计算字段名称，有两点除外：计算字段名称能隐含或显示多个主题，也可以使用名称的复数形式。但是，如有可能，最好使用单数形式。
- **过滤器**：指示该视图显示记录的过滤条件。被测试的字段和测试所用的表达式都要记录。

❖ **注意**：当填写视图规范表中的计算字段表达式和过滤器部分时，应使用最为熟悉的表达式。如有必要，在 RDBMS 中实施该数据库时，将会进行修改。

分别为每个视图填写视图规范表，视图规范表应与视图示意图一一对应。这两者都用于充分记录视图。图 12.16 所示为 CUSTOMER CALL LIST 完整的视图规范表。（记住，这个视图已加入 CustState 字段。）

[illegible]

图 12.16 完整的 CUSTOMER CALL LIST 视图规范表

评审每个视图的文档记录

定义和记录视图的任务完成后,就可以再次评审所有的视图。目的是确保每个视图提供的信息质量上乘。评审每个视图时,要注意以下几点。

- **确保正确定义视图。**考虑一下视图所提供的信息。你是否为所要求的信息选择了正确的视图类型?定义视图所用基表是否恰当?视图结构中是否包含了所有必要字段?又或者其中是否只包含了必要的字段?
- **确保所创建的计算字段适用于该视图。**这些计算字段是否提供了相关且有意义的信息?它们是否有助于提升视图展示数据的方式?
- **确保过滤器检索要求的记录。**首先,该视图需要过滤器吗?如果需要,你希望该视图具体显示那些记录呢?另外,你觉得这个过滤器能发挥相应作用吗?
- **最重要的是,确保每个视图都有对应的示意图和规范表。**最终,在 RDBMS 中实现数据库时,视图示意图和规范表将发挥重要作用。

案例分析

迈克数据库的工作已经接近尾声。此次,你与迈克和他的员工会谈,是为了确定是否需要为数据库建立视图。会谈设置的议程如下:

1. 回顾设计过程中汇编的记录。
2. 逐个回顾早期收集的各种样本。
3. 检查数据库中各个表所表示的主题。
4. 分析表关系。

5. 回顾和研究业务规则。

随着会谈的推进,你们发现了多个需要定义的视图,包括 **PREFERRED CUSTOMERS** 视图和 **VENDOR PRODUCT COUNT** 视图。第一个视图将提供所有拥有“优先”级别的顾客姓名和电话,第二个视图提供每位供应商

供应各种产品总量的信息。

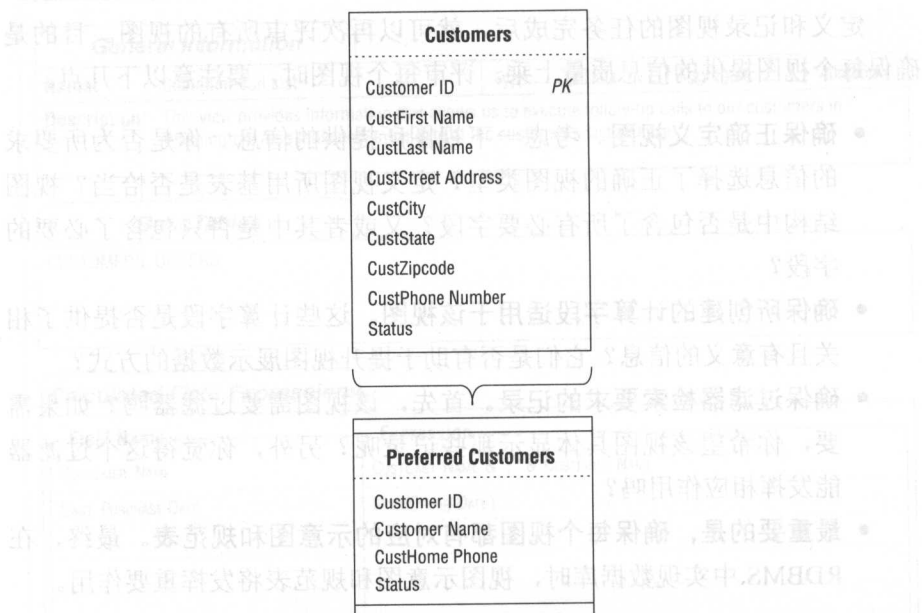


图 12.17 PREFERRED CUSTOMERS 视图的示意图

PREFERRED CUSTOMERS 视图基于 CUSTOMERS 表，使用 Customer ID、CustFirst Name、CustLast Name、CustHome Phone 和 Status 字段组成视图的结构。不过，在构建这个视图之前，迈克向你询问是否能把姓和名一起显示出来。你给出了肯定的回复。于是，你又创建了一个计算字段 Customer Name，将两个字段串接起来；这个字段将取代 CustFirst Name 和 CustLast Name。图 12.17 所示为 PREFERRED CUSTOMERS 视图的示意图。

画出视图示意图后，你记录了过滤该视图数据所用的表达式：

Status = "Preferred"

然后，完成了 PREFERRED CUSTOMERS 视图的规范表。图 12.18 所示即为其结果。

General Information		
Name:	Preferred Customers	Type: <input checked="" type="checkbox"/> Data <input type="checkbox"/> Aggregate <input type="checkbox"/> Validation
Description:	This View provides the names and phone numbers of our Preferred customers. We use this information in support of the services we provide to these customers.	

Base Tables	
CUSTOMERS	

Calculated Field Expressions	
Field Name	Expression
CUSTOMERNAME	CUSTFIRSTNAME & " " & CUSTLASTNAME

[illegible]

图 12.18 PREFERRED CUSTOMERS 视图的规范表

现在，以 VENDORS 和 PRODUCTS 作为基表定义 VENDOR PRODUCT COUNT 视图。使用 VENDORS 表中 Vendor Name 字段展示供应商的名称。接下来，创建计算字段 Product Count，显示每位供应商供应商品的总量。下列即为该字段用于计算总量的表达式：

Count(ProdName)

如图 12.19 所示即为 VENDOR PRODUCT COUNT 视图示意图。

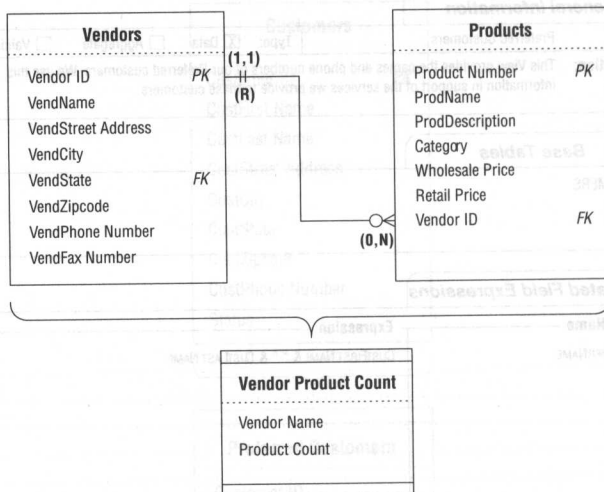


图 12.19 VENDOR PRODUCT COUNT 视图示意图

随后，你确定这个视图的一个过滤器是多余的。据此，完成了图 12.20 所示的视图规范表。

然后，对迈克数据库识别的每个视图重复这一过程。

VIEW SPECIFICATIONS																			
General Information																			
Name: Vendor Product Count	Type: <input type="checkbox"/> Data <input checked="" type="checkbox"/> Aggregate <input type="checkbox"/> Validation																		
Description: This view tells us how many products are supplied by each vendor. This information will help us determine which vendors we might need to drop.																			
Base Tables																			
VENDORS, PRODUCTS																			
Calculated Field Expressions																			
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Field Name</th> <th style="text-align: left; padding: 2px;">Expression</th> </tr> </thead> <tbody> <tr> <td style="padding: 2px;">PRODUCTCOUNT</td> <td style="padding: 2px;">Count(PRODUCTNAME)</td> </tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> </tbody> </table>	Field Name	Expression	PRODUCTCOUNT	Count(PRODUCTNAME)															
Field Name	Expression																		
PRODUCTCOUNT	Count(PRODUCTNAME)																		
Filters																			
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Field Name</th> <th style="text-align: left; padding: 2px;">Condition</th> </tr> </thead> <tbody> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> <tr><td style="padding: 2px;"> </td><td style="padding: 2px;"> </td></tr> </tbody> </table>		Field Name	Condition																
Field Name	Condition																		

图 12.20 VENDOR PRODUCT COUNT 视图的视图规范表

小结

本章开头介绍了视图的定义，即视图是不包含数据或存储数据的虚拟表。视图的用途体现在这三点：同时使用多个表中的数据、有效提升数据完整性，以及有效保持数据安全和保密。

接着，本章讨论了三种视图：**数据、聚合和验证**。每种视图都可以基于一个或多个表，其他视图，或两者的结合。每次访问一个视图，RDBMS 都会使用其基表的大部分最新数据，重建和重新植入它。文中也提到，多表视图的基表之间必须存在关系（从而使该视图的信息有效且具有意义），这些关系的特征通过视图体现出来。另外，大多数视图都可以修改，所有的修改都将通过视图，反映到其基表中。虽然**验证视图**和验证表的运作方式相同，但是前者具有显著的优势。例如，验证视图能包含多个表中的数据。

然后，又继续讨论了确立视图。首先，介绍了与用户和管理人员一起识别机构对视图的要求时需要注意的几点事项。接着，讨论了如何定义视图，和通过创建**视图示意图**将视图记录在文档中。最后，还提到了如何从基表中选取字段并指定到视图中。

接下来，讨论的重点是如何在视图中使用计算字段。计算字段有助于提供相关信息，也能提升视图展示数据的方式。计算字段在聚合视图中尤为关键，每个计算字段都通过使用一个表达式得出其显示的值。此外，还介绍了在视图中运用过滤器，以检索和显示特定记录集合。仅当一个视图符合对其中一个或多个字段施加的条件时，该视图才会显示给定记录。具体做法就是，将每个条件都设计成一个表达式，使用表达式测试特定字段的值。

本章末尾讨论了视图规范表。其中，提到了使用视图规范表记录视图的特征，还介绍了视图规范表的各个项。

思考题

1. 为什么说视图是虚拟表？
2. 说出视图重要的两个原因。
3. 说出设计数据库逻辑结构时可以定义的视图类型。
4. 每次访问数据视图（或其他类型的视图）时，RDBMS 都会做些什么？
5. 对视图数据所做修改类型由什么决定？
6. 为定义多表数据视图，必须满足什么要求？
7. 为什么数据视图本身不包含主键？
8. 聚合视图的用途是什么？
9. 对一组数据最常用的聚合函数是什么？
10. 什么是组合字段？
11. 判断题：聚合视图中的数据可以修改。
12. 验证表和验证视图的区别是什么？
13. 识别视图要求时，应考虑哪两点？
14. 计算字段应在什么情况下使用？
15. 如何定义视图，使它只显示科幻书籍？
16. 为什么必须为数据库中每个视图制订视图规范表？

小结

顾李思

第 13 章

评审数据完整性

当你排除了不可能的，剩下的无论是什么、多么叫人难以置信，都一定是真相。

——夏洛克·福尔摩斯

《四签名》

本章内容

为什么要评审数据完整性

评审和改进数据完整性

汇编数据库文档

大功告成

案例分析——总结

小结

现在，到了数据库设计的最终环节。自从这个过程开始以来，已经先后完成了许多任务，现分列如下：

- 了解关系数据库模型的优势并与其他数据库模型进行比较
- 为新数据库制订宗旨
- 为新数据库定义任务目标
- 对原有数据库进行完整分析

- 识别机构信息要求
- 为所有适宜的表定义结构
- 为每个表指定主键
- 为每个字段建立字段说明
- 建立表关系
- 定义和建立业务规则
- 定义所有适宜的视图
- 建立整体数据完整性

无论从哪点来看,新数据库已经完成。不过,最好对数据库的整体数据完整性执行最后一次评审。

为什么要评审数据完整性

鉴于整个设计过程可谓一丝不苟,对数据完整性更是全神贯注,读者可能会问,为什么最后还要评审一次数据完整性。答案很简单:要确保投入如此多精力才得以建立的数据完整性臻于完美。从上文中可知,完整性哪怕有一丝裂缝,都可能导致数据的不一致或信息的不准确。无论多么难以置信,都可能有所遗漏。如果你能确认数据库设计优良,从中获得的内心的安宁绝对物超所值。

❖注意:记住,种瓜得瓜,种豆得豆!

评审和改进数据完整性

如果采取模块化的方法,即依次评审整体数据完整性的每个部分:表层次、字段级和关系层次完整性,以及业务规则,评审数据完整性就会变得十分简单。如果之前严格遵循本书中介绍的设计方法,就应该不会碰到多少问题。下一节将简单列出开展评审时应注意的几个点,其中还提到了可参考的章节,以防遇到任何问题。

表层次完整性

为了确保表层次完整性规范正确,对每个表进行评审,保证所有表符合以下几点要求。

- 表中无重复字段。
- 表中无计算字段。
- 表中无多值字段。
- 表中无复合字段。
- 表中无重复记录。
- 表中每个记录都通过一个主键值识别。
- 每个主键都满足主键的要素。

如果违反以上任意要求,可运用第6章“分析现有数据库”、第7章“建立表结构”和第8章“键”中的技巧和概念解决问题。

字段级完整性

达到以下几点要求,就能确保字段级完整性规范正确:

- 确保每个字段符合理想字段的要素。
- 确保为每个字段定义一套字段说明。

参见第9章“字段说明”,可解决字段级完整性出现的问题。

关系层次完整性

检查每个表的关系,确保关系层次完整性规范正确。完成以下任务,就能实现这个层次完整性:

- 建立恰当关系。
- 定义合适的删除规则。
- 准确识别每个表的参与类型。
- 为每个表确立适宜的参与度。

如发现任何关系方面的问题，可参见第 10 章“表关系”进行解决。

业务规则

确保以下几点，就能获得完善的业务规则。

- 每条规则施行有意义的限制。
- 为规则选择合适的类别。
- 正确定义和建立每条规则。
- 修改适宜的字段说明元素或表关系特征。
- 建立适宜的验证表。
- 为每条规则建立业务规则规范表。

如遇到任何与业务规则相关的问题，参见第 11 章“业务规则”，查找相应的技巧予以解决。

视图

尽管视图并不与数据完整性的任何部分直接相关，但是仍然应该评审所有视图的结构。检查每个视图时，要确保它满足以下条件。

- 每个视图都包含必要的基表，以提供所要求的信息。
- 为每个视图指派合适的字段。
- 每个计算字段提供相关信息或改善该视图展示数据的方式。
- 每个过滤器返回适宜的记录集合。
- 每个视图都有视图示意图。
- 每个视图示意图具有对应的视图规范表。

如遇到任何视图方面的问题，参见第 12 章“视图”予以解决。

待整个评审过程完成，你就能确认数据库的结构完善，数据库中的数据一致有效，以及从数据库中检索的信息准确无误。

汇编数据库文档

整个数据库设计过程产生了许多列表、规范表和示意图，它们被用于记录数据库设计的各个方面。应该将它们都汇集到一起，存入中央仓库，最好是整齐有序地归入计算机文件夹中。这个设计仓库应包含以下文档：

- 字段表列表
- 字段说明表
- 计算字段列表
- 表结构示意图
- 关系示意图
- 业务规则规范表
- 视图示意图
- 视图规范表

读者可能认为另外两种材料也应添加进去，分别是整个设计过程中编辑的记录和分析阶段收集的样本。这两部分材料可以作为文档末尾的附录。

所有这些组成了数据库逻辑设计的一整套文档资料。这套文档资料的重要性体现在以下三个方面。

1. 为数据库结构提供完整的记录。在这套文档资料中，可以查找到数据库逻辑设计各个方面的记录。此外，参考这套资料，几乎可以解答与该数据库相关的所有问题。
2. 为实现过程应该如何创建数据库提供一整套规范和操作说明。这套资料与建筑师的设计蓝图相似：它指明了如何构建该数据库。另外，它也识别了建立该数据库所需的完整性。因为数据库设计并不针对特定 RDBMS，实现数据库的人员在实际实现该数据库的方式上有充分的自由。
3. 在实现过程中，如有必要修改数据库结构，这套设计文档资料就可以用于判定任何修改的影响和结果。在决定对数据库结构进行修改

前，应提前预判结果。通过参考文档资料，你可以确保修改不会对数据库结构造成不利影响。

大功告成

既然已经完成对数据完整性的评审，并汇编了该数据库的所有文档。逻辑数据库的设计过程也就此宣告结束。你可以放心，新数据库设计得当，其实现过程也将顺利进行。现在，可以为下一个客户服务，开始下一个数据库的设计工作了。

案例分析——总结

这是你与迈克和他的员工的最后一次会谈。目标是评审他的数据库及其完整性。尽管你坚信不会出现任何问题，但还是想再对这个数据库进行一次质量控制的评审。

在会谈期间，你们对数据库结构的每个部分依次评审，确保它们符合相应的各种要素。然后，评审整体数据完整性的各个部分，确保表层次、字段级和关系层次完整性，以及业务规则建立正确规范。最后，收集整个设计过程产生的所有文档材料。将这些文档材料整理到一个文件夹中，并转交给迈克，声明他的数据库已经圆满完成。迈克称赞了你的杰作，并向你表示诚挚的谢意，同时许诺支票将在本月 15 日前寄到。你也向迈克和他的员工表示感谢，之后告别，转身离去。马克目送你离开，随即一个念头浮上脑海：

“现在，如果我请你来为我实现数据库……”

小结

本章开头列举了数据库设计过程开始以来的各项任务。然后，讨论了应该再次评审整体数据完整性的原因。接着，简要讨论了评审整体数据完整性各个部分时应注意的事项。结尾部分讨论了整个设计过程中收集的文档资料的重要性。

第 3 部分

其他数据库设计事项

[illegible]

第 14 章

设计不当——禁忌事项

错误总有源头。

——切萨雷·帕韦斯

本章内容

- 平面文件设计
- 电子表格设计
- 基于数据库软件设计数据库
- 最后一点想法
- 小结

读者也许会疑惑，为什么这一章出现在本书末尾而不是开头呢？原因很简单：既然已经学会如何设计规范的数据库，自然也能够体会到数据库设计不当的危害。此外，读者已经能够独立探究特定设计欠佳的原因——审视该设计，立即找出结构中的问题。你还拥有了解决这些问题所需的知识。

本章将介绍导致数据库结构不佳的三种最为常见的设计方法。由此所阐发的讨论将力求简洁，因为这只是为了说明应避免的设计方法。至此，解决数据库设计不当的方法显然是遵照上文所述的完整设计过程。

平面文件设计

这种设计方法（有时又称为“一表式”设计）已存在多年，常见于为非关系型数据库管理系统的实现而设计的数据库。检查图 14.1 所示的结构，就会发现平面文件设计存在着许多问题。

Table Structures		
Customer Orders		
Order Number	Customer Phone	Item 2 Extension
Order Date	Item 1	Item 3
Ship Date	Quantity 1	Quantity 3
Order Amount	Price 1	Price 3
Sales Rep Name	Item 1 Extension	Item 3 Extension
Customer Number	Item 2	
Customer Name	Quantity 2	
Customer Address	Price 2	

图 14.1 平面文件结构示例

这个示意图表示一个表的结构（想想数据库中其他表的结构是怎样的！）。不难发现，这个结构势必会造成冗余数据和数据前后不一致的问题。此外，它也缺乏数据完整性。读者也许早就注意到，这个结构还有其他一些问题。

- **复合字段：**Sales Rep Name 包含推销员的姓和名，Customer Name 包含顾客的姓和名，Customer Address 包含顾客的街道地址、所在城市、州以及邮编。
- **计算字段：**Order Amount 字段包含的值极有可能需要手动计算，如果该顾客订购三项以上就尤为必要。几个 Item # Extension 字段也都需要手动计算。给定 Item # Extension 字段的值是，相关 Quantity # 字段的值乘以相关 Price # 字段的值得出的结果。（例如，Item 3 Extension

= Quantity 3 × Price 3)

- **无用重复字段：**与特定项相关的每个字段都是重复的。例如，Item 1、Item 2 和 Item 3 字段都是无用重复字段。
- **无真正的主键：**这个表中没有可唯一识别单一记录的字段或字段组。Order Number 字段不是这个表的主键。如果顾客的订购项超过三个，就必须使用相同的订单号输入另一个记录。
- **这个表表示多个主题。**这个表表示三个主题：顾客、订单和项（从另一个角度看，它也表示推销员这个主题）。

既然了解了规范数据库设计的要素，想必能够避免这样的设计。

电子表格设计

如果使用得当并且符合设计意图，那么电子表格确实是一个不错的工具。例如，对于复杂数学计算和统计分析，电子表格设计就十分适用。不过，与多数人的看法相反，电子表格并不能制作优良的关系数据库。如果机构需要收集、存储、维护和操作各种数据，就要选用合适的工具，设计和实现真正的数据库。例如，思考图 14.2 中的电子表格。

	A	B	C
1	Store 100 (344-0029)		Store 103 (554-2993)
2	Manager: Mike Hernandez		Manager: Katie Christian
3	Asst. Mgr: Bob McNeal and		Asst. Mgr: Terri Sharpe
4	Suzi Thompson		
5	Store 101 (433-4872)		Store 104 (773-1837)
6	Manager: Abe Hernandez		Manager: Gay Wayne
7	Asst. Mgr: Steve McMahn		Asst. Mgr: Barbara Clark
8			and Tim Ennis
9	Store 102 (433-4872)		Store 105 (344-2883)
10	Manager: Susan Black		Manager: Carmine Aguilar
11	Asst. Mgr: Diana Price		Asst. Mgr: Lee Sampson

图 14.2 典型电子表格“数据库”示例

这个电子表格用于记录一家小型零售连锁店的各个经理的数据。从图中可知,这种方法也存在问题。

- **重复字段:** 电子表格中的每个字段都是重复字段。如果根据外表判断,每个实例都有三个基本字段: Store Number、Manager Name 和 Assistant Manager Name。
- **复合字段:** 每个字段都有两个值。第一个字段存储店面编号和电话号码,第二个字段存储经理的姓和名,第三个字段存储副经理的姓和名。
- **多值字段:** Assistant Manager 字段是多值字段,因为特定店面被指派了多个副经理。
- **难以使用:** 面向数据的任务在 RDBMS 程序中能轻易执行,但是在电子表格中却更麻烦且比较耗时。例如,创建一个列表,将每位店面经理的姓名和电话号码包含进去,就需要花费一定量的时间。

看到这些由简单电子表格“数据库”引发的问题,可以想象这类复杂数据库将带来怎样严重的问题。如果当前使用一个电子表格作为数据库,就可以将它从中移出,遵照整个数据库设计过程进行处理,再在合适的 RDBMS 中实现,这样可以提升该数据库的质量、速度和多功能性。

摒弃电子表格视图思维定式

当开始应对真正的数据库和 RDBMS 时,就必须摒弃电子表格视图的思维定式。这意味着必须认清一个事实:观察数据的特定方式已经不再适用,即不能再使用典型的电子表格布局。例如,图 14.3 所示的典型电子表格报表。

使用数据库无法产生具备这种布局的报表,这种报表展示的正是电子表格存储的数据。不过,数据库可以将它存储在一个表的四个独立的字段中。图 14.4 所示为使用相同数据生成的数据库报表的示例。虽然数据库演示与电子表格演示不同,但是却一样条理清晰。

记住一点,必须调整你对处理数据库中数据的看法。最终,利用真正的数据库存储和使用数据的许多优势将会突显出来,这是使用电子表格所不能比的。数据库不仅能赋予你对数据完整性和数据一致性、有效性更多的控制,

也能为检索数据提供几近无数种方式，让你获得广泛多样的信息。

Branch Stores		
Bellevue		
Store 118 Manager: Katherine Ehrlich	Store 201 Manager: Kevin Swanson	Store 211 Manager: George Chavez
Redmond		
Store 27 Manager: Mark Rosales	Store 75 Manager: Chris Warren	Store 322 Manager: Steve Horst
Seattle		
Store 105 Manager: Caroline Cole	Store 187 Manager: Julia Black	Store 200 Manager: Alan Jacob

图 14.3 典型电子表格报表示例

Branch Stores	
Bellevue	Seattle
Store 118 Manager: Katherine Ehrlich	Store 105 Manager: Carmen Aguilar
Store 201 Manager: Kevin Swanson	Store 187 Manager: Julia Black
Store 211 Manager: George Chavez	Store 200 Manager: Alan Jacob
Redmond	
Store 27 Manager: Mark Rosales	
Store 75 Manager: Chris Warren	
Store 322 Manager: Steve Horst	

图 14.4 典型数据库报表示例

基于数据库软件设计数据库

RDBMS 并不能为设计数据库提供任何基础或步骤，即使是创建数据库

的原因也无从了解——它只提供实现设计所需的工具。相反,规范的数据库设计方法能提供正确定义数据的必要原则和理论基础。

许多人不经意间就掉进了设计数据库的陷阱,认为仅仅基于其实现过程所用的 RDBMS 软件就可以设计数据库。在许多情况下,这种看法的产生是由于他们对特定 RDBMS 的熟悉和熟练或者所在公司和机构早就使用了某种 RDBMS。出于以下原因,应避免这种不明智的做法。

- 可能基于主观判断对设计做出决策。例如,你也许决定不对给定关系施加参与度,因为你认为该 RDBMS 不提供施加参与度的方法。
- 由于疏忽大意,让 RDBMS 指定的数据库设计完全背离了机构信息要求。这种情况通常发生在发现 RDBMS 对数据库的特定方面(比如字段说明和关系特征)只提供有限支持的时候。
- 对 RDBMS 的了解程度限制设计水平。例如,你也许决定不执行关系特征,原因仅仅是不知道其具体做法。
- 设计受自身对 RDBMS 的熟练程度所限。熟练程度影响实现数据库的各个方面的效率,比如字段说明和业务规则。
- 使用这种方法设计数据库通常导致结构设计不当,数据完整性不足,以及数据不一致和信息不准确。在 RDBMS 中定义数据库貌似轻松。这样创建的数据库也许能够运作,但是极有可能埋下隐患。
- 最后,你熟知且喜爱的 RDBMS 也许不能适应机构数据库要求。

应该在始终不考虑任何 RDBMS 的前提下,设计数据库的逻辑结构。这样,才更有可能设计出结构完善的数据库,因为你可以专注于机构的信息要求。设计完成后,就可以清楚地判断如何实现数据库(单用户应用程序、客户端/服务器、基于网络等)以及使用哪种 RDBMS 便于实现。

最后一点想法

本书作者执教数据库设计多年,一直指导人们如何使用各种 RDBMS 软件程序,最后发现一个有趣的现象:熟悉规范数据库设计的基本原则的人,往往对所使用的 RDBMS 及其工具有更好的认识。笔者认为,这是由于熟悉

数据库设计的人能理解 RDBMS 提供特定工具的原因,也清楚可以(且应该)如何运用这些工具。由于这一点以及本书中提到的其他原因,学习和理解优秀的数据库设计技巧对读者而言有着显著的优势。设计数据库并不仅限于本书描绘的这种方法,但是它却是最直接、最可靠、也是最容易的。

小结

本章将关系数据库设计与其他设计做了对比。首先,谈到了平面文件设计。这种方法存在大量致命的错误,应坚决禁止。然后,又提到了电子表格设计,这种方法作用非常有限。最后,讨论了使用 RDBMS 软件设计数据库。这种设计方法取决于个人对相应软件的熟悉和熟练程度,隐藏着不小的风险。与优秀的数据库设计方法不同,在 RDBMS 中设计数据库并不能为设计规范的结构提供必要原则和理论基础。从短期来看,RDBMS 软件产品看似不错。但是,正如本书所讨论的,长此以往,这种设计方法终究会出现问题。

第 15 章

打破规则

自然从不违反自身规律。

——莱昂纳多·达芬奇

本章内容

何种情况下可以打破规则

记录行动

小结

本书始终提倡遵循规范的数据库设计技巧，而且上文也已多次提及其中的原因。不过，首先你应该使用优秀的设计方法，确保数据库的完整性。其重要性想必不言而喻。现在，知道了数据完整性设计不当的后果，最重要的就是遵守这些规则。

何种情况下可以打破规则

唯有两种特定情况才允许打破规范的数据库设计规则。除非这两种情况不可避免，否则应该使用规范的数据库设计技巧设计数据库。

设计分析型数据库

第 1 章“关系数据库”中已经提到，分析型数据库存储和追踪历史和时间依赖性数据。这种数据库往往在一些表结构中包含了计算字段。这些字段中的大多数所用的表达式用于在指定时刻记录特定一组数据的状态；其他的则存储聚合函数的结果。

读者可能早已从上述语句中推测出，这种数据库违反了规范的数据库设计，因为它的表中包含计算字段（参见第 7 章“建立表结构”）。在这个特定的例子中，这种违反之所以可以接受正是由于分析型数据库使用数据的方式。本书推荐先遵循规范设计数据库，然后经过慎重的考虑之后，再打破规则。对此，应该根据具体情况，分析这一做法的必要性。

❖注意：设计分析型数据库的方法与本书所介绍的方法完全不同。如果认定机构需要一个分析型数据库，本书强烈推荐你查找一本相关的好书，学习如何正确设计这种数据库。

提升处理性能

这依然是人们认为必须打破规则的最常用理由。每次 RDBMS 处理多表查询或复杂报表花费了过多的时间，许多人就会认为解决这一问题的办法就是改变基础表结构。例如，他们会让你修改一个表，以使它包含该查询或报表需要的每个字段。虽然这种修改确实提高了 RDBMS 处理该查询或报表的速度（特别是在老式系统中），但是它也引入了一些新问题，比如无用重复字段和冗余数据。这显然不是理想的解决方案，它违背了规范的数据库设计。

不幸的是，现实生活并不能尽如人意，所以有时会发现自己在提升处理性能和遵守规范设计原则之间做出抉择。

值，还是不值

当你真正为这一困境停下来思考片刻时，就会立即意识到问题的关键在于性能；而是数据完整性。每次为了性能（或任何其他原因）而打破规则，势必会给数据完整性造成问题。那么，就得问自己：是否值得为处理性能得

以提升的感觉，付出损害数据完整性的代价？如你所知，草率修改给数据结构造成的恶果，最终会像池塘中的水波蔓延到整个数据库。下列问题仅为其中的一些。

- **数据不一致**：这是向表中引入无用重复字段的结果。读者（或应用程序）的责任是确保这些字段中的数据保持同步。如果修改了特定重复字段中的值，就必须保证对剩余重复字段也做出相同的修改。
- **冗余数据**：冗余数据也是向表中引入无用重复字段的结果。当编辑一个字段中的特定值，而该字段中正好包含冗余数据，就必须确保对该值的每个例子进行相同的修改。
- **受损的数据完整性**：打破规则往往会损害到整体数据完整性的一个或多个部分，比如表层次完整性和关系层次完整性。读者（或应用程序）的责任是竭尽所能修补受损的完整性。
- **不准确的信息**：如果数据库存在上述任何问题，就难以提供准确信息。

提升性能首选其他方式

如果为了提升处理性能，仍然想使用这种做法，那就当作最后手段。不过，采取这些措施之前，尽量先选择其他方式。可以考虑以下方案。

- **提升或更新计算机硬件**。成本已不再是问题，所以这是提升处理性能最为简单的方式。比如更快的 CPU、更大的内存、更快更高效的磁盘驱动器，以及更符合打印要求的打印机等，这些都会有助于极大地缩短 RDBMS 处理复杂查询或报表所需的时间。
- **调整操作系统软件**。确保计算机的操作系统优化至最佳性能。对于联网计算机和服务器软件，这一点尤为重要。更改配置选项设置能大幅提升综合处理性能。对操作系统所做修改一般取决于所用的操作系统，所以必须参阅说明资料，判断该如何修改。
- **评审数据库结构**。务求数据库设计得当。其意义十分重大。设计欠佳的数据库实际上导致了处理性能不佳。
- **评审数据库的实现过程**。检查当前数据库在 RDBMS 中的实现过程。确保充分利用 RDBMS 的性能，以及尽可能高效和完善地定义数据库。

● **评审数据库的应用程序。**对此必须认真检查。应用程序是否编写正确？是否充分利用了 RDBMS 提供的工具？应用程序的组成部分是否定义准确？在某些情况下，报表也许由于设计不当而打印较为缓慢。对于设计和生成相同报表，也许存在更为有效的方法。查询也会因为定义不规范而运行缓慢。确保每次查询定义正确且采取最为高效的方式。

如果认为必须摒弃规范数据库设计技巧，就要认真考虑你的处境。正如上文所提到的，如果是设计分析型数据库，放弃规则就是可以接受的。但是，本书仍然强烈建议采取规范完整的方法设计数据库，无特殊情况绝不放松要求。

记录行动

如果排除了所有其他选项，坚持认为需要打破规则，就必须记录打破的每条规则和采取的每一步行动！尤其是记录所做的改变，因为这样就能迫使你考虑其后果，也为修改数据库结构提供记录。假如以后发现这种修改并不能带来显著的好处，就可以根据记录的指引恢复到修改之前。

以下为需要记录的项。

- **打破规则的原因：**最常见的两个原因是，提升处理性能和减少打印复杂报表所需时间。无论是什么原因，都必须完整清楚地加以描述。
- **违反的设计原理：**记录对数据库设计的更改，如果发现性能没有大幅提升，也能保证之后可以恢复原样。例如，可以记下正在修改表的结构。
- **修改的数据库部分：**指示将要修改的特定字段、表、关系或视图。再次提醒，这一信息对于恢复到修改前的原样十分重要。
- **所做的修改：**一旦认定需要修改的项，就要准确记录对该项所做的修改。例如，如果需要修改一个关系，就要准确记录对其特征的修改。
- **对数据库和应用程序预期的影响：**对数据库所做的任何修改都会影响到所有相关的终端用户应用程序。例如，修改特定表的结构会影响到

数据完整性、视图结构、数据输入表单和基于该表建立的报表（部分或全部），还有与该表相关的程序编码。必须确保记录所有影响。

将这些记录添加到数据库汇编的文档资料中。即使修改之后又恢复原样，这些记录也可以引为前车之鉴。

小结

本章开篇即介绍了打破规范数据库设计方法的两种情况。其中一种就是设计分析型数据库，在这种情况下，打破规则是可以接受的；否则，应首先按照正规的方法设计数据库，然后再慎重决定是否要打破特定的规则。打破规则最为常见的理由是提升处理性能。尽管这一理由并不能让人满意，但是在一些情况下你必须考虑这一无奈之举。

然后，本章继续讨论了提升处理性能的替代措施，比如提升或更新硬件以及评审数据库的实现过程。因此，首先应该尽可能地提升性能，打破规范设计技巧只能当作最后手段。本章最后列举了打破规则之后需要记录的事项。

结束语

我不是导师，只是个同路人。你问路在何方？我就指着前方——你和我的前方。

——萧伯纳

笔者始终认为，并非行家才能设计规范的数据库。只要掌握了大量的常识性知识，任何人都能完成这项相对简单的任务。严格遵循优秀数据库设计方法，就能设计出完善可靠的数据库结构。

到现在，读者已经掌握了设计关系数据库所需的知识和技巧，学会了如何定义必要的结构，建立表关系，以及实现各个层次的数据完整性。如果遇到设计不规范的结构，也知道如何加以改善。

学习数据库设计是一个持续的过程。可以根据需要学习设计特定的数据库类型，也可以将之作为一项职业，还可以当作终生研究的课题。无论如何，都会遇到一个不可避免的事实：学得越多，越发明白自己的无知。但是不要泄气。钻研任何主要学科都会产生这种感受，比如音乐、艺术、哲学以及航空航天学。

我真诚希望，读者能像我编写本书一样，带着喜欢去阅读。我知道这个领域的多数技术类书籍读来有些枯燥，所以绞尽脑汁想要添加几丝幽默，特别是在访谈和会谈的对话中。如果你们认为其中对话比较逼真，则表明观察力相当敏锐。它们都是从我多年来与客户的访谈和会面中提取的。

作为临别赠言，不妨给大家一点建议：**勤学不辍**。不要害怕或拒绝学习

新知识。通过学习，才能接触到新思想、新概念和新观念。学习可以促进人与人之间的分享和交流，开拓每一个人的眼界。

学习如千里之行，始于足下。读完本书，你就迈出了第一步。接下来，你将继续前行，学习数据库管理的其他方面。

本书就此结束，但是你的行程才刚刚开始……

附录

题 关系型数据库管理系统, 简称 RDBMS, 是专门用于创建、维护、修改和查询关系数据库的软件程序。

A 案例

第 4 部分

案例题答案

第 2 章

附录

1. 使用数据库设计工具的最佳时间是在正式数据库逻辑结构之前。
2. 正确。设计对于数据库的一致性、完整性和准确性至关重要。
3. 数据库模型是数据库设计的基础。
4. 关系数据库模型以集合论为基础。
5. 数据库设计方法学的优点是：
 1. 提供设计完整的数据库结构所需的技巧。
 2. 提供设计完整的数据库结构所需的技巧。
 3. 提供设计完整的数据库结构所需的技巧。
 4. 提供设计完整的数据库结构所需的技巧。
 5. 提供设计完整的数据库结构所需的技巧。
 6. 提供设计完整的数据库结构所需的技巧。
 7. 提供设计完整的数据库结构所需的技巧。
 8. 提供设计完整的数据库结构所需的技巧。
 9. 提供设计完整的数据库结构所需的技巧。
 10. 提供设计完整的数据库结构所需的技巧。
 11. 提供设计完整的数据库结构所需的技巧。

附录 A

思考题答案

第 1 章

1. 现在主要使用的两种数据库是**操作型**和**分析型**数据库。
2. 分析型数据库存储**静态**数据。
3. 正确。操作型数据库主要用于联机事务处理场景。
4. 关系数据库模型出现之前，常用**分层**和**网状**两种数据库模型。
5. 在父/子关系中，一个父表可与一个或多个子表相关联，但单一子表只能与一个父表相关联。
6. **集合结构**是网状数据库中建立和表示一个关系的一种透明结构。
7. 关系模型基于两个数学分支：**集合论**和**一阶谓词逻辑**。
8. 关系数据库将数据存储于关系中，用户则将关系视为表。
9. 关系数据库中的关系类型包括：**一对一**、**一对多**和**多对多**。
10. 关系数据库中通过使用结构化查询语言（SQL）检索数据。
11. 关系数据库具有这些优点：**内置多层次完整性**；**逻辑和物理数据独立于数据库应用程序**；**有保障的数据一致性和准确性**；**便捷数据检索**。

12. 关系数据库管理系统, 或 RDBMS, 是一个用于创建、维护、修改和操作关系数据库的软件程序。
13. 对象关系模型通过加入各种面向对象的元素和特征, 比如类、封装以及继承, 拓展了关系数据库模型。
14. 数据仓库让机构可以访问存储在任何关系和非关系数据库中的数据。

第 2 章

1. 使用 RDBMS 程序设计工具的最佳时间是设计完数据库逻辑结构之后。
2. 正确。设计对于数据的一致性、完整性和准确性至为关键。
3. 数据库设计不当的最恶劣后果是信息不准确。
4. 关系数据库模型以集合论和一阶谓词逻辑为基础, 使得关系数据库结构健全并能确保信息准确。
5. 学习设计方法学的优点是:
 - a. 提供设计完善的数据库结构所需的技巧。
 - b. 提供一整套技巧, 引导你逐步完成设计过程。
 - c. 帮助你将失误和设计重复控制在最低水平。
 - d. 简化设计过程, 缩短设计数据库所需时间。
 - e. 帮助你更充分、有效地理解和运用 RDBMS 软件。
6. 正确。理解数据库设计有助于更有效地使用 RDBMS 程序。
7. 优秀设计的目标包括:
 - a. 数据库支持要求的和特殊的信息检索。

- b. 表的构建正确且高效。
 - c. 数据完整性在字段、表和关系三个层次上实施。
 - d. 数据库支持相应于机构的业务规则。
 - e. 数据库能够适应未来的拓展。
8. 数据完整性有助于确保数据结构及其值始终准确有效。
9. 运用优秀设计技巧的好处有：
- a. 数据库结构易于修改和维护。
 - b. 数据易于修改。
 - c. 信息易于检索。
 - d. 终端用户应用程序易于开发和建立。
10. 错误。走捷径、缩减部分设计过程，就无法达成优秀健全的设计。

第 3 章

1. 术语的重要性主要体现在：
- a. 用于表达和定义关系数据库模型的特殊思想和概念。
 - b. 用于表达和定义数据库设计过程本身。
 - c. 用于讨论关系数据库或 RDBMS。
2. 术语的四大类分别是与值相关、与结构相关、与关系相关以及与完整性相关。
3. 数据库中存储的值是数据。信息则是处理或查看时，通过一定处理手段使之具有意义且有用的数据。
4. null 表示一个缺失或未知的值。

5. null 的主要缺点是会对数学运算造成不利影响。
6. 表是数据库的主要结构。
7. 三种表分别是数据表、联系表和验证表。
8. 视图是由数据库中一个或多个基表的字段构成的虚拟表。
9. 键是用于识别表中记录的逻辑结构, 索引是用于优化数据处理的物理结构。
10. 两表之间存在的三种关系分别为一对一、一对多和多对多。
11. 关系的特征可以用三种方式进行描述: 表之间存在的关系类型、每个表的参与类型和每个表的参与度。
12. 字段说明表示一个字段的元素。
13. 字段说明包括三类元素: 一般、物理和逻辑元素。
14. 数据完整性是指数据库中数据库的有效性、一致性和准确性。
15. 数据完整性的四种类型分别是字段级、表层次、关系层次和业务规则。

第 4 章

1. 因为遵循完整的设计过程有助于确保健全的结构和数据完整性。
2. 正确。结构完整性的层次与设计过程的规范程度成正比。
3. 宗旨识别的是数据库的用途。
4. 任务目标是表述用户可利用数据库中数据执行的总任务的语句。
5. 设计过程第二阶段汇编的字段和计算列表构成了公司的基本数据要求。
6. 根据设计过程的第一阶段中编辑的任务目标和第二阶段收集的数据要求, 确定表代表的各个主题。

7. 错误。为每个字段建立字段说明是在数据库设计过程的第三阶段。
8. 通过使用主键或联系表，建立一种关系中的各表之间的逻辑联系。
9. 机构认识和使用其数据的方式决定了一系列写入数据库的限制和要求。
10. 如有必要，设计和实施验证表可支持特定业务规则。
11. 通过与用户和管理人员访谈，分别了解他们如何使用数据，从而确定写入数据库的视图类型。
12. 完成完整的数据库设计过程之后，就可以在 RDBMS 程序中实现逻辑数据库结构。

第 5 章

1. 访谈之所以重要，是因为它为你（数据库开发者）和服务对象提供了重要交流机会。另外，访谈也能有效确保设计工作取得成功，为数据库结构的设计提供关键信息。
2. 对许多人访谈会出现的问题是，一些参与者会因为参与人数的增多而变得拘束，产生威胁感。
3. 对用户和管理人员分别访谈的主要原因是，每个群体对机构的整体看法不同，对机构如何使用日常数据也有不同见解。
4. 错误。访谈中通常使用开放式问题。
5. 你应该引导参与者做出完整、描述性的回答。
6. 访谈最重要的一条指南是始终掌握对访谈的控制。
7. 宗旨指的是对数据库具体用途的概括。
8. 优秀的宗旨具有这些特征：
 - a. 表述清楚。

- b. 简洁扼要。
 - c. 无明确描述具体任务的词汇或语句。
9. 错误。一个任务目标不能描述多个任务。
10. 根据回答中显示或暗含的信息可以得出任务目标。
11. 当一个任务目标得到规范定义且你和服务对象都能理解该任务目标时，这个任务目标就宣告完成。

第 6 章

1. 分析现有数据库的目标是为了确定：
- a. 机构使用哪种数据。
 - b. 机构如何使用其数据。
 - c. 机构如何管理和维护其数据。
2. 错误。不应该采用现有数据库结构作为新数据库结构的基础。
3. 遗留数据库是存在和被使用不低于五年的数据库。
4. 分析过程包括这三步：
- a. 评审收集数据的方式。
 - b. 评审信息呈现的方式。
 - c. 与用户和管理人员开展访谈。
5. 分析过程中应评审的计算机软件程序包括文字处理器、电子表格、数据库和网页。
6. 得到数据收集和呈现的样本后仍需访谈的原因是：
- a. 访谈能为之前评审过程中收集的样本提供细节信息。

- b. 访谈能提供机构使用数据的方式的信息。
 - c. 访谈有助于定义初始字段和表结构。
 - d. 访谈有助于定义信息要求。
7. 开放式问题侧重于具体主题, 封闭式问题侧重于特定主题的具体细节。
8. **确定主题技巧**让你能从参与者对某问题的回答中找出主题。
9. 使用**确定特征技巧**确定特定主题的具体特征。
10. 错误。应该分别对用户和管理人员进行访谈。
11. 必须确认的三种基本信息要求为**当前信息要求**、**附加信息要求**和**未来信息要求**。
12. **初始字段列表**表示机构基本数据要求, 它包含了数据库定义的核心字段集合。
13. 初始字段列表中的每一项都应具有独特的名称, 确保该特征只在列表中出现一次。
14. **值列表**是指特定特征的可接受值的范围, 它常常实施一条特定的业务规则。
15. 计算字段存储字符串连接或计算表达式的结果作为它的值。应该将计算字段从初始字段列表中移动到专门的计算字段列表中。

第7章

- 1. 使用**初始表列表**为新数据库识别和创建表。
- 2. 使用初始字段列表有助于定义表的原因是, 初始字段列表中的字段可能暗含数据库需要记录的主题。
- 3. 当主题列表和初始表列表中有名称不同、表示主题相同的项时, 应

该选取最能代表该主题的名称，将之作为该主题的唯一标识。

4. 最终表列表为数据库中每个表提供名称、类型和描述。
5. 创建表名称的指南分别是：
 - a. 表名称应独特且富有内涵。
 - b. 表名称应明白无误地展现其主题。
 - c. 表名称应尽量精简。
 - d. 避免使用描述物理特征的词语。
 - e. 避免使用缩略语。
 - f. 避免使用专有名称或其他过多限制输入数据的词语。
 - g. 避免使用隐含或显示多个主题的名称。
 - h. 使用复数形式。
6. 编写表描述的指南分别是：
 - a. 对表进行准确定义。
 - b. 解释该表对机构的重要性。
 - c. 描述务求简明扼要。
 - d. 避免提及具体操作信息，比如该表使用的方式和适用场合。
 - e. 不同表描述之间保持独立。
 - f. 避免使用示例。
7. 通过判断哪个字段最能代表表的特征，从而将该字段分配到最终表列表的相应表中。
8. 创建字段名称的指南分别是：

- a. 字段名称应独特且有意义。
 - b. 字段名称应简明扼要, 准确描述字段所代表的特征。
 - c. 字段名称应求精简, 避免拖沓冗长。
 - d. 切勿使用缩略语, 慎用缩写词。
 - e. 切勿使用混淆字段名称含义的词语。
 - f. 避免使用隐含或显示多个特征的名称。
 - g. 使用名称的单数形式。
9. 设计欠佳的字段会产生重复数据和冗余数据。
10. 确保该字段满足理想字段要素, 就能解决字段异常。
11. 理想字段的要素包括:
- a. 代表表主题的鲜明特征。
 - b. 仅包含一个值。
 - c. 无法分解为更小的元素。
 - d. 不含计算值或串联值。
 - e. 在整个数据库结构中独一无二。
 - f. 即使出现在多个表中, 其主要特性始终保持不变。
12. 解决多值字段或无用重复字段后, 剩余的冗余数据是可以接受的。
13. 一般而言, 解决多值字段遵循这三个步骤:
- a. 从相应表中移除该字段, 以之作为基础创建一个新的表。
 - b. 使用原表中的一个字段(或一组字段)作为关联字段, 建立起原表和新表的联系。

- c. 为新表制定合适的名称, 编写恰当的描述, 并将该表添加到最终表列表中。
- 14. 表中需要使用重复字段的唯一情况是, 该字段被用于建立两表之间的关系。
- 15. 确保每个表满足理想表的要素, 就能精简表结构。
- 16. 理想表的要素包括:
 - a. 表示单个主题, 包括物体或事件。
 - b. 拥有一个主键。
 - c. 不含复合字段或多值字段。
 - d. 不含计算字段。
 - e. 不含无用的重复字段。
 - f. 冗余数据量保持最低水平。
- 17. 子集表是表示特定数据表的从属主题的表。

第 8 章

- 1. 键重要的原因有:
 - a. 确保表中每个记录正确识别。
 - b. 有助于建立和实施各种完整性。
 - c. 用于建立表关系。
- 2. 键的四种主要类型分别是候选键、主键、外键和非键。
- 3. 候选键的用途是唯一标识表主题的单一实例。
- 4. 候选键的要素包括:

- a. 不得为复合字段。
 - b. 必须包含唯一值。
 - c. 不得包含 null 值。
 - d. 其值无论部分或整体都不可选。
 - e. 包含定义唯一性所需的最少字段。
 - f. 其值必须是识别表中每个记录的独特和唯一方式。
 - g. 其值必须是给定记录中每个字段值的唯一方式。
 - h. 除非特殊情况，否则不得修改其值。
5. 正确。候选键可以由多个字段组成。
 6. 是，一个表可以拥有多个候选键。
 7. 创建一个字段，其唯一用途是当作候选键，则将该字段称为人造候选键。当表中无“自然出现”的候选键时，就创建人造候选键。
 8. 表中最重要的是主键。
 9. 主键的重要性体现在这些方面：
 - a. 主键字段是整个数据库结构中识别其所在表的唯一方式。
 - b. 主键值同时是相应表中和整个数据库中特定记录的唯一识别方式。它也有助于防止重复记录。
 10. 检查相应表中可用候选键，选择其中一个作为主键。
 11. 主键的要素包括：
 - a. 不得为复合字段。
 - b. 必须包含唯一值。
 - c. 不得包含 null 值。

- d. 其值无论部分或整体都不可选。
 - e. 包含定义唯一性所需的最少字段。
 - f. 其值必须是识别表中每个记录的独特和唯一方式。
 - g. 其值必须是给定记录中每个字段值的唯一识别方式。
 - h. 除非特殊情况, 否则不得修改其值。
12. 选定主键之前, 必须确保该字段是给定记录中每个字段值的唯一识别方式。
13. 替换键是未被选为该表主键的候选键。
14. 建立表层次的完整性, 就能确保:
- a. 表中无重复记录。
 - b. 主键是表中识别每个记录的唯一方式。
 - c. 每个主键值都是唯一的。
 - d. 主键值不是 null。
15. 评审初始表结构是由于这些原因:
- a. 确保合适主题在数据库中被表达。
 - b. 确保表名称和表描述规范且准确易懂。
 - c. 确保字段名称规范且准确易懂。
 - d. 核实所有字段已指定给每个表。

第 9 章

1. 字段说明的重要性体现在这几个方面:
- a. 字段说明有助于建立和实现字段级完整性。

- b. 为每个字段定义字段说明能提升整体数据完整性。
 - c. 定义字段说明能获得对数据的性质和用途的完整认识。
 - d. 字段说明构成了数据库的“数据字典”。
2. 字段级完整性能保证这些优势：
 - a. 字段特性和用途明确，它所出现的表都得到正确识别。
 - b. 整个数据库中字段定义一致。
 - c. 字段的值一致且有效。
 - d. 修改、比较和操作字段值所运用的方法有了明确界定。
 3. 字段说明中的三类元素分别是一般元素、物理元素和逻辑元素。
 4. 三类说明的名称分别是独特、通用和可复制。
 5. 创建规范的字段描述，能让你（和机构内部所有人都）认真思考该字段中存储的数据的性质。
 6. 数据类型元素指示该字段所存储数据的性质。
 7. 字符支持元素指示用户可以输入给定字段值的字符类型。
 8. 显示格式元素控制屏幕上展示或文档中打印出来时该字段值的外观。
 9. 字段说明中指示的键的类型包括非键、主键、替换键和外键。
 10. 错误。null 不表示空白，而表示缺失或未知值。
 11. 值的范围元素指示该字段所有可能的有效值。
 12. 编辑规则指定用户什么时候向该字段输入一个值，以及是否能修改该值。
 13. 允许的比较元素指示用户从该字段检索信息时，可对该字段的给定值运用的比较类型。

14. 值的表达式是某种形式的元素, 涉及字段值、文字值或两者的结合, 它返回的单一值可用于比较运算。
15. 通用说明适用于数据库中作为其他字段模板的字段。

第 10 章

1. 关系的重要性体现在这些方面:
 - a. 对于存在逻辑联系的两表, 可在两表之间建立连接。
 - b. 有助于进一步改进表结构和将冗余数据减至最少。
 - c. 这种机制能实现同时从多个表中提取数据。
2. 三种关系分别是一对一、一对多和多对多。
3. 多对多关系引发的问题最多。
4. 处理多对多关系可能碰到这些问题:
 - a. 从其中的表检索信息将变得烦琐, 甚至困难。
 - b. 一表中将包含大量冗余数据。
 - c. 两表中都存在重复数据。
 - d. 难以插入、更新和删除数据。
5. 自引用关系是给定表的记录之间存在的一种关系。
6. 识别数据库中表间关系应从创建所有表的矩阵入手。
7. 使用关系型和情境型问题有助于识别现有关系。
8. 使用 1:N 速记符号表示一对多关系。
9. 使用三种关系类型的定义对应的公式, 判断矩阵中两表之间的正式关系类型。

10. 复制位于关系“一”端的主键，将之加入位于“多”端的表中，作为其外键，就建立了一个一对多关系。
11. 正确。如果一个表具有自引用关系，从该表中检索信息就将变得烦琐且有些困难。
12. 建立多对多自引用关系与建立表间多对多关系相似，也要使用一个联系表。
13. 确保每个外键满足外键的要素，从而改进外键。
14. 外键字段说明中必须修改一般元素和逻辑元素两个类别。
15. 当发出请求，要求删除关系中父表的给定记录，删除规则就会根据这一要求决定 RDBMS 应做出的反应。
16. 参与类型分为强制和随意。
17. 参与度指示给定表必须与相关表单一记录关联记录的最小数量和容许与相关表单一记录关联记录的最大数量。
18. 待验证该关系建立规范及其特征设置合理后，就实现了关系层次完整性。

第 11 章

1. 业务规则表示对数据库特定方面实施的某种形式的限制，比如特定字段说明中的元素和给定关系的特征。
2. 两种主要的业务规则分别是面向数据库和面向应用程序。
3. 不能。数据库逻辑设计过程无法建立面向应用程序业务规则施加的限制。
4. 面向数据库业务规则的两个类别是字段特有和关系特有。
5. 字段特有业务规则就是向特定字段的字段说明中的元素施加限制的

业务规则。

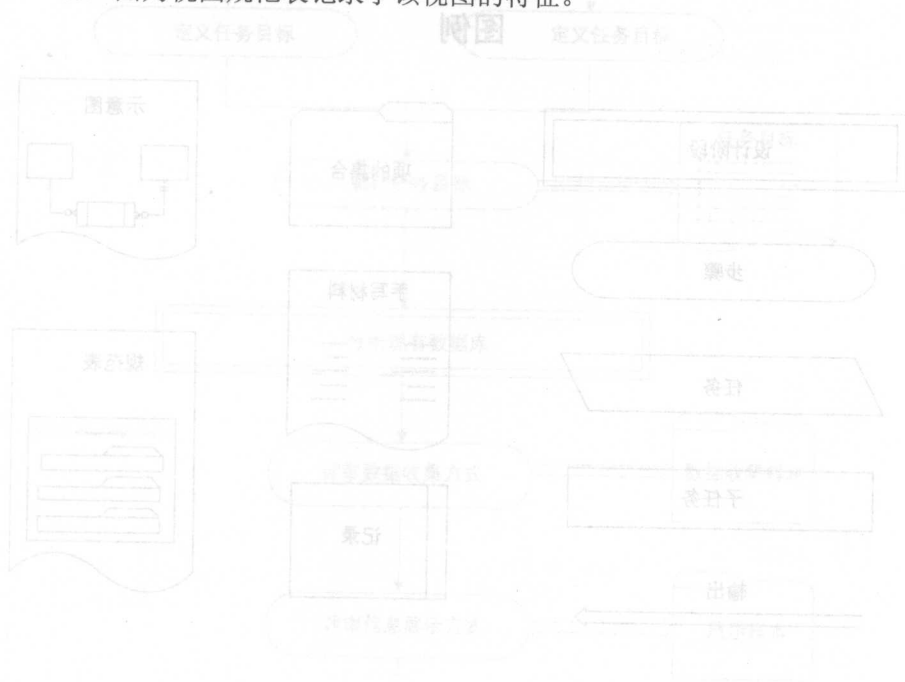
6. 执行三个操作中任一项都会测试业务规则：向相应表中插入一个记录或字段中插入一个条目；删除相应表中一个记录或字段中的一个值；更新一个字段值。
7. 填写某业务规则的规范表，就可以将该业务规则记录在文档中。
8. 业务规则规范表具有这些优点：
 - a. 可以将所有面向数据库业务规则使用文档记录。
 - b. 可以将所有面向应用程序业务规则使用文档记录。
 - c. 为记录所有业务规则提供一种标准方法。
9. 采取的措施部分的用途是指示字段说明或关系示意图所做修改。
10. 验证表（又称为查找表）存储专门用于实现数据完整性的数据。
11. 验证表通常（但并不始终）包含两个字段：第一个作为主键，将用于帮助实施数据完整性；第二个只是一个非键字段，用于存储数据库中另一个字段所要求的一组值。
12. 业务规则对给定字段值的范围施加限制，验证表则执行该限制。
13. 为了确保每个业务规则规范表记录的规则建立规范，规范表中所有项填写清楚，所以评审所有完成的业务规则规范表。

第 12 章

1. 视图被称为虚拟表的原因是，它只是从其基表中提取数据，本身并不存储数据。
2. 视图的重要性体现在这些方面：
 - a. 可同时处理取自多个表中的数据。

- b. 反映最新的信息。
 - c. 根据个人或群体的特定需求定制。
 - d. 有助于实施数据完整性。
 - e. 用于保密和安全用途。
3. 设计数据库逻辑结构时可以定义的视图类型包括：**数据、聚合和验证视图**。
4. 每次访问视图，RDBMS 都会使用其基表中的最新数据，重建和重新植入这个视图。
5. 字段说明和业务规则决定了能对相应视图的数据做出的修改类型。
6. 定义多表数据视图必须满足的唯一要求是，用于创建这种视图的表之间存在关系。
7. 数据视图不包含其自身的主键，因为它不是一个表；真正的表存储数据，并需要一个主键作为每个记录的唯一识别方式。
8. 聚合视图的用途是展示特定数据集聚合所产生的信息。
9. 数据集最常用的聚合函数是总和、平均值（算术平均值）、最小值、最大值和计数。
10. 组合字段是聚合视图中的数据字段，它将给定值的多个实例组合成该值的一个实例。
11. 错误。聚合视图中的数据不可修改，因为它完全是由组合字段和计算字段组成的。
12. 验证表和验证视图的区别在于它们的构造，验证表本身存储数据，而验证视图则从其基表中提取数据。
13. 识别视图要求时要遵循这几点：
- a. 和代表们一道回顾你的记录。

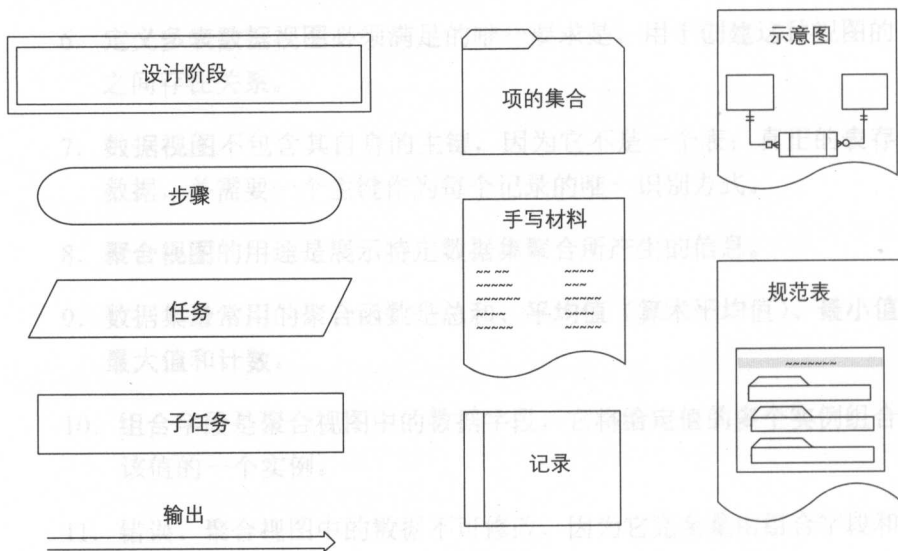
- b. 回顾设计早期阶段收集的数据输入、报表和演示样本。
 - c. 检查表及其所表示的主题。
 - d. 分析表关系。
 - e. 研究业务规则。
14. 当计算字段能提供相关且有意义的信息或有助于提升视图使用数据的方式时, 应该使用计算字段。
15. 为了使视图只显示科幻书籍, 可在该视图中应用相应过滤器。
16. 因为视图规范表记录了该视图的特征。

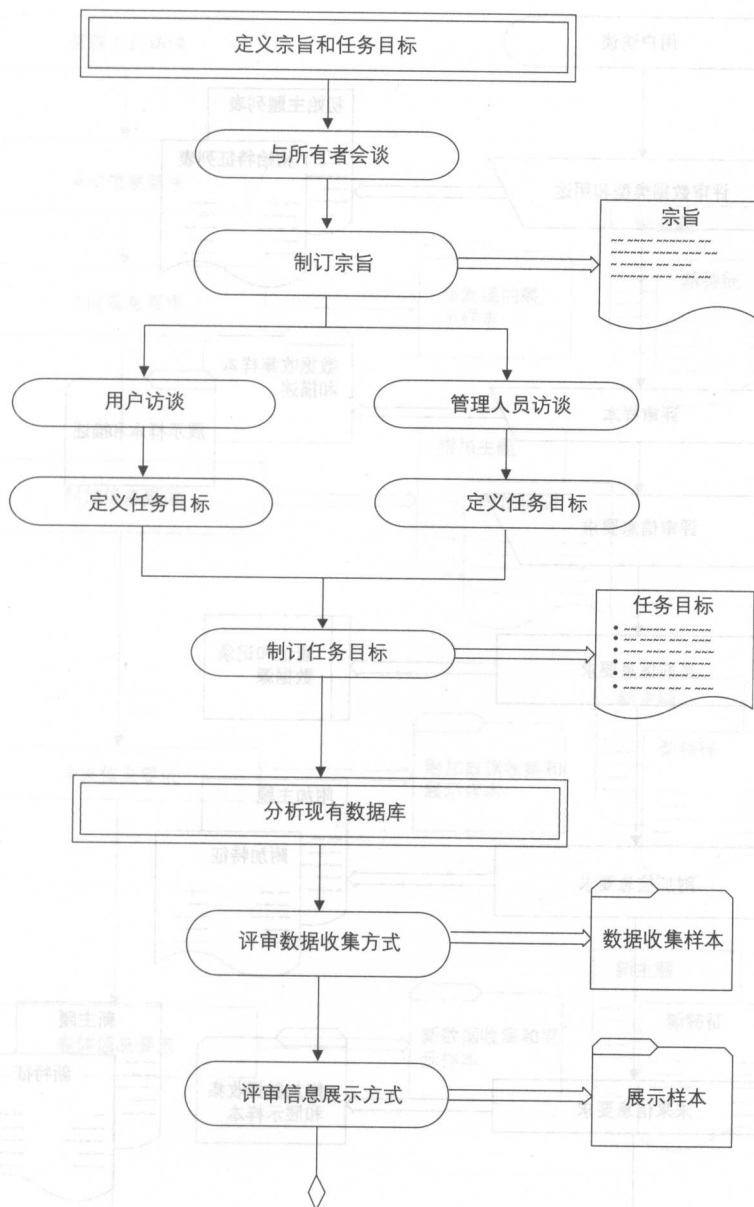


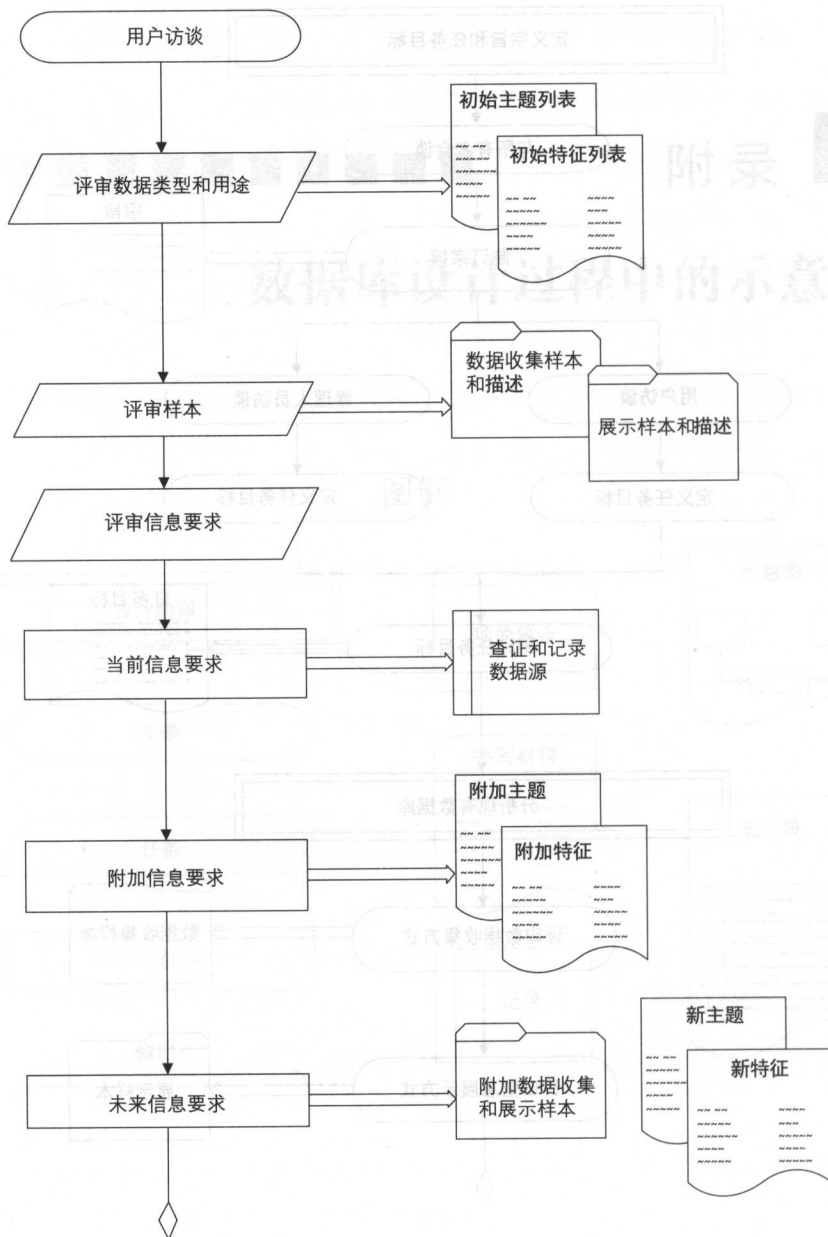
附录 B

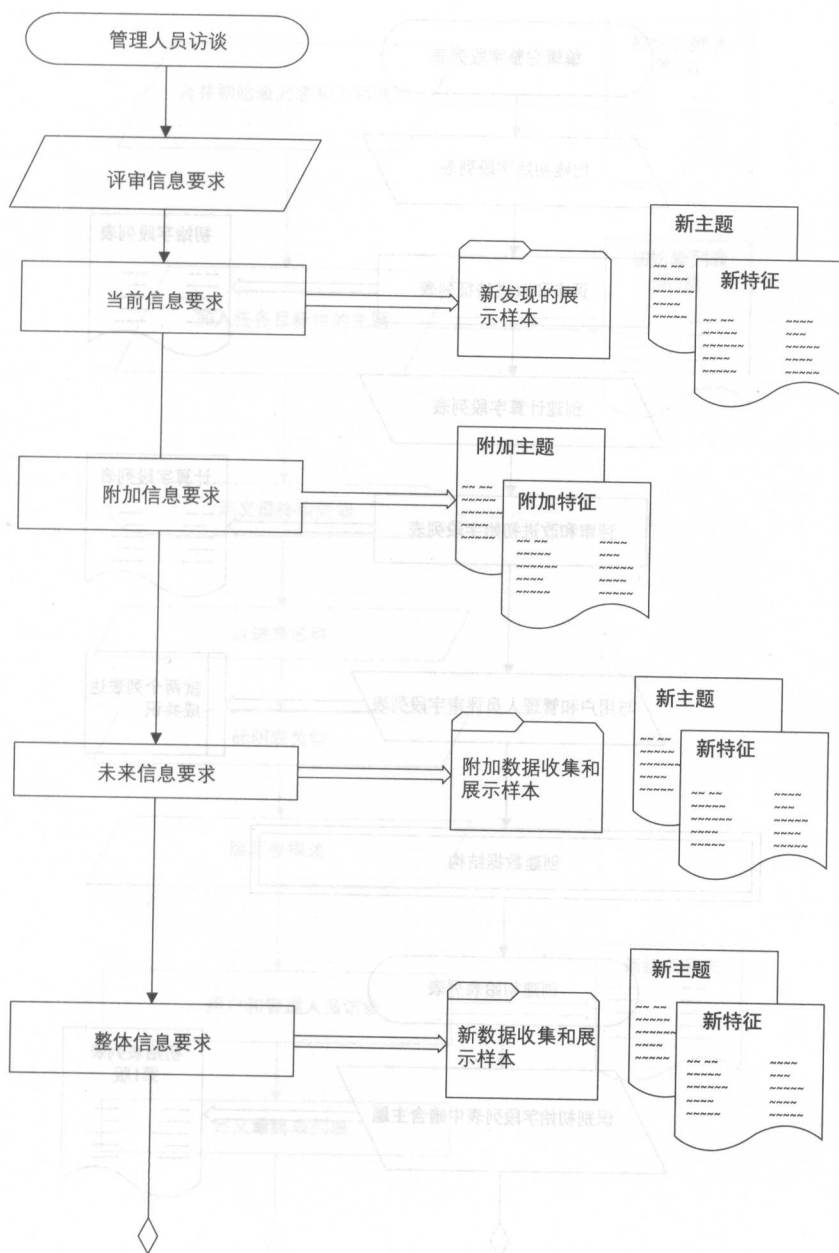
数据库设计过程中的示意图

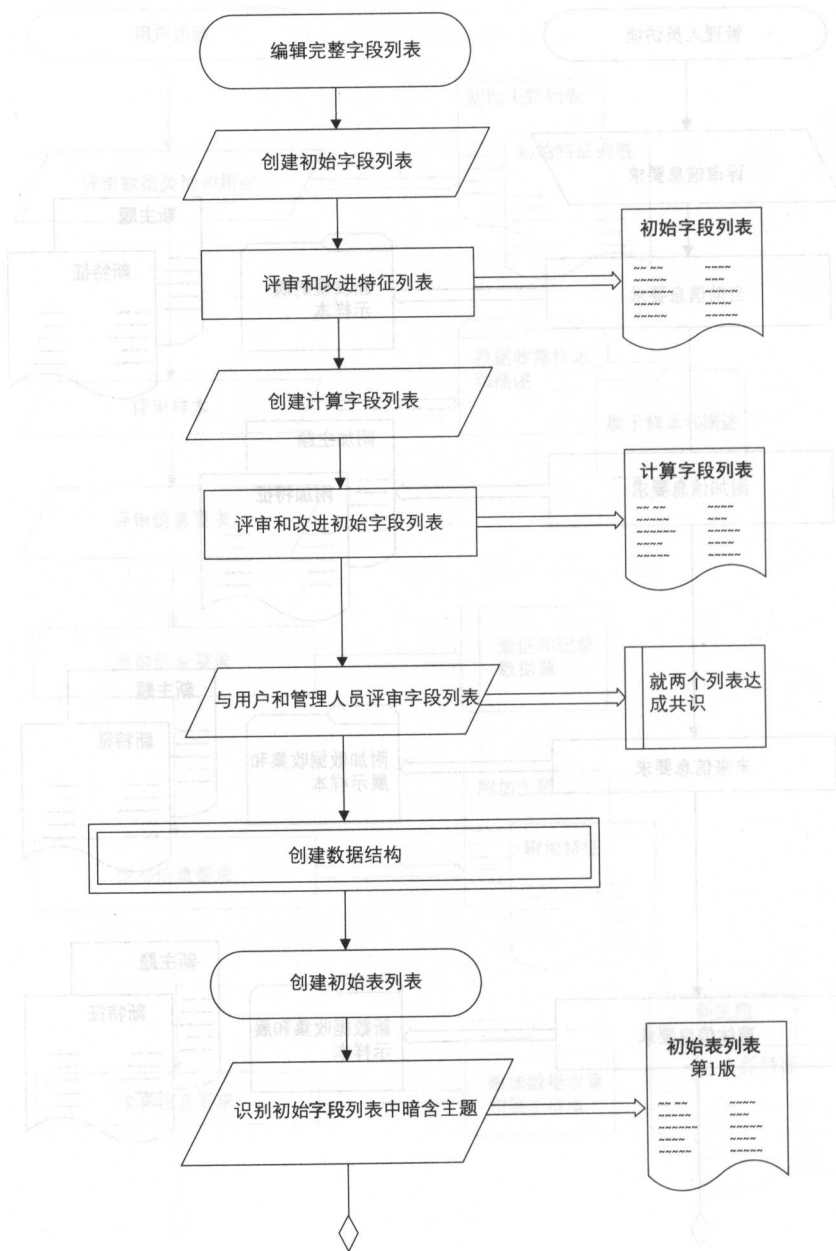
图例

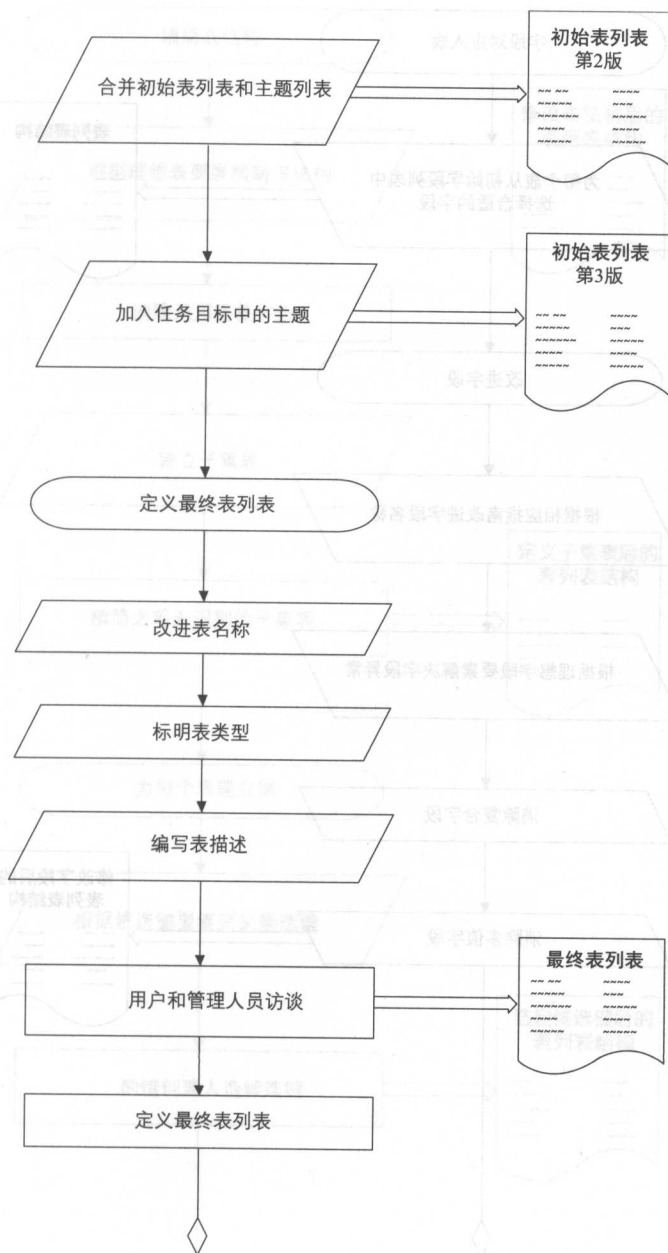


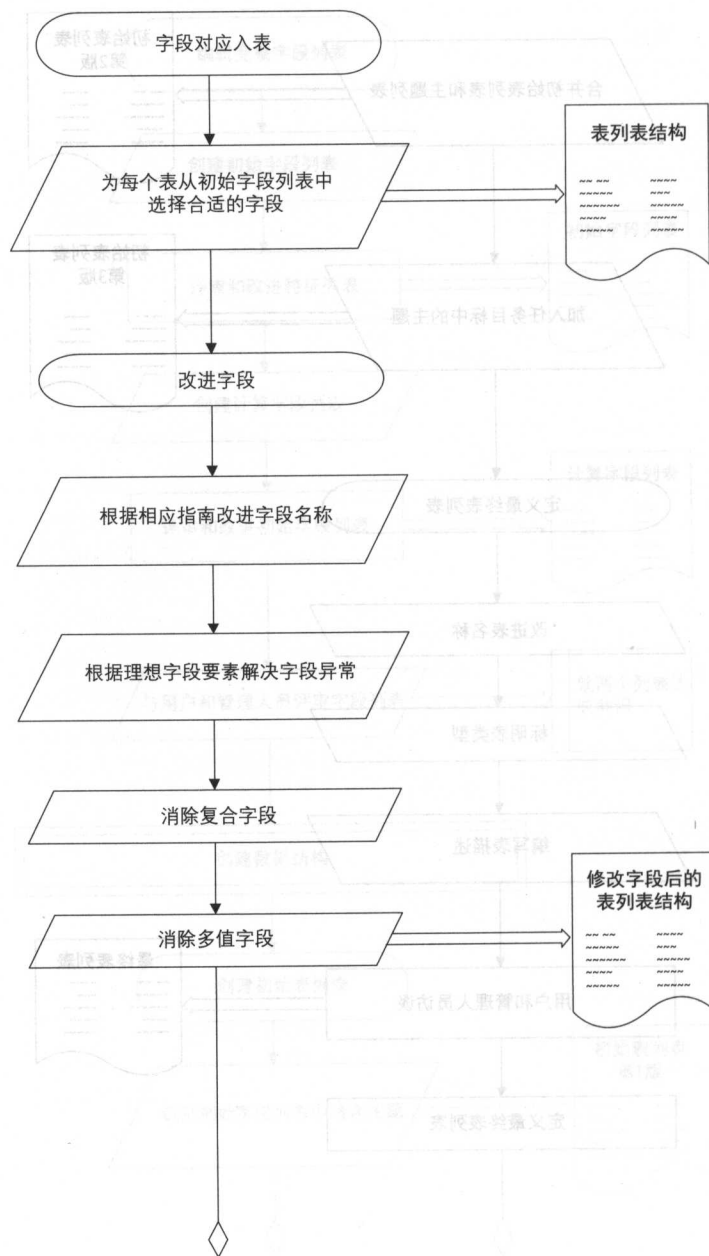


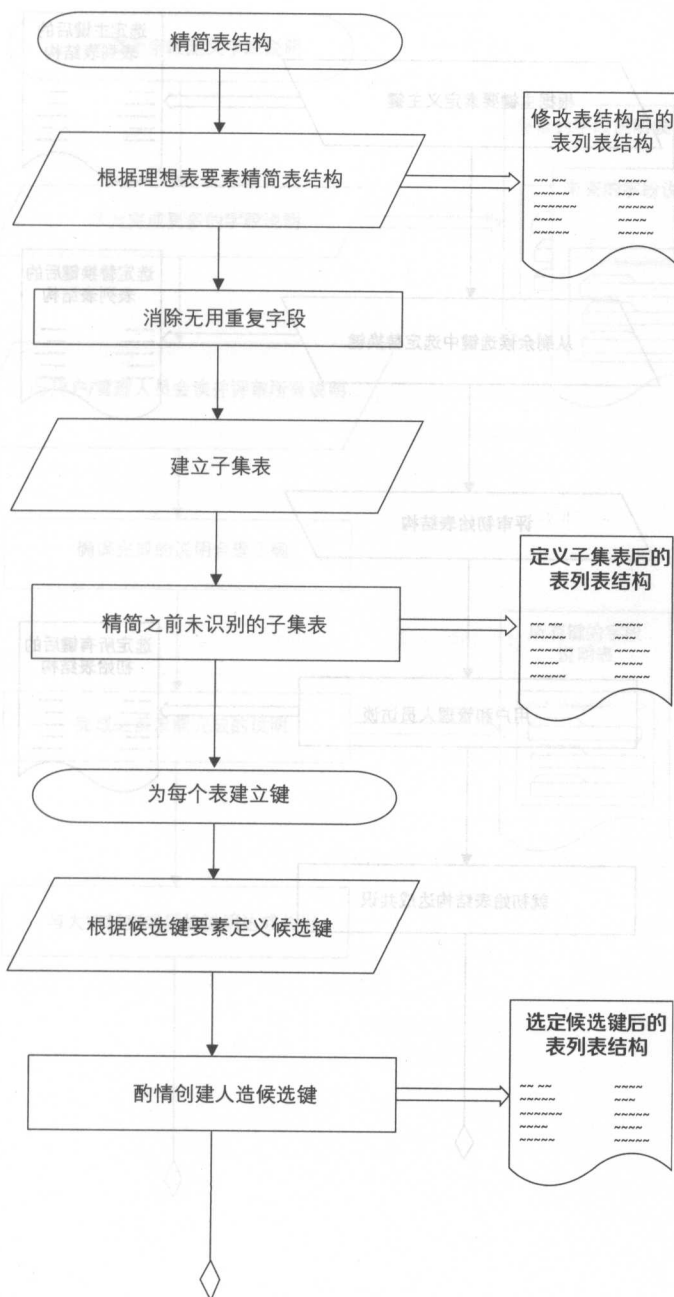


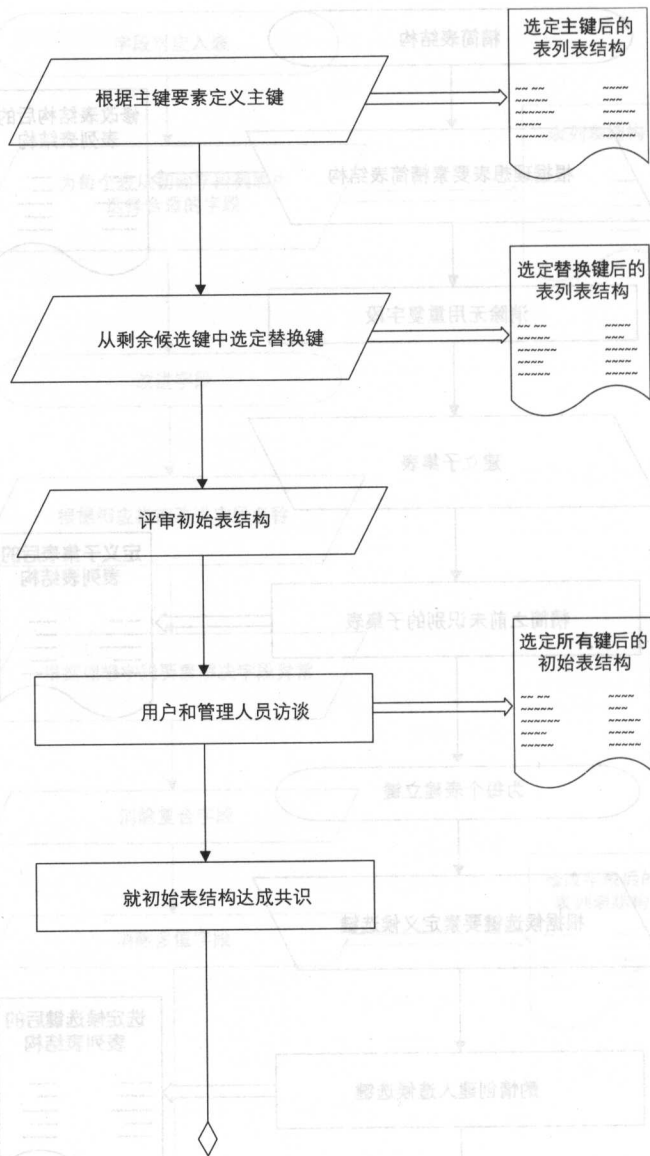


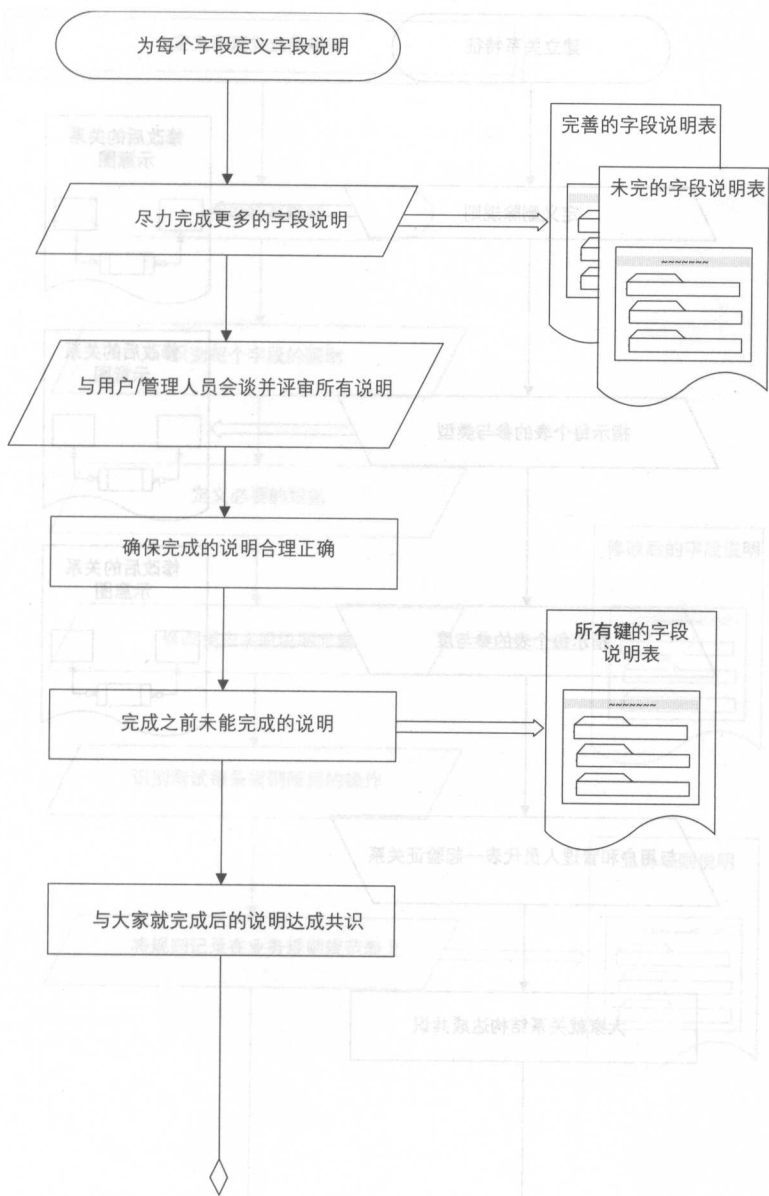


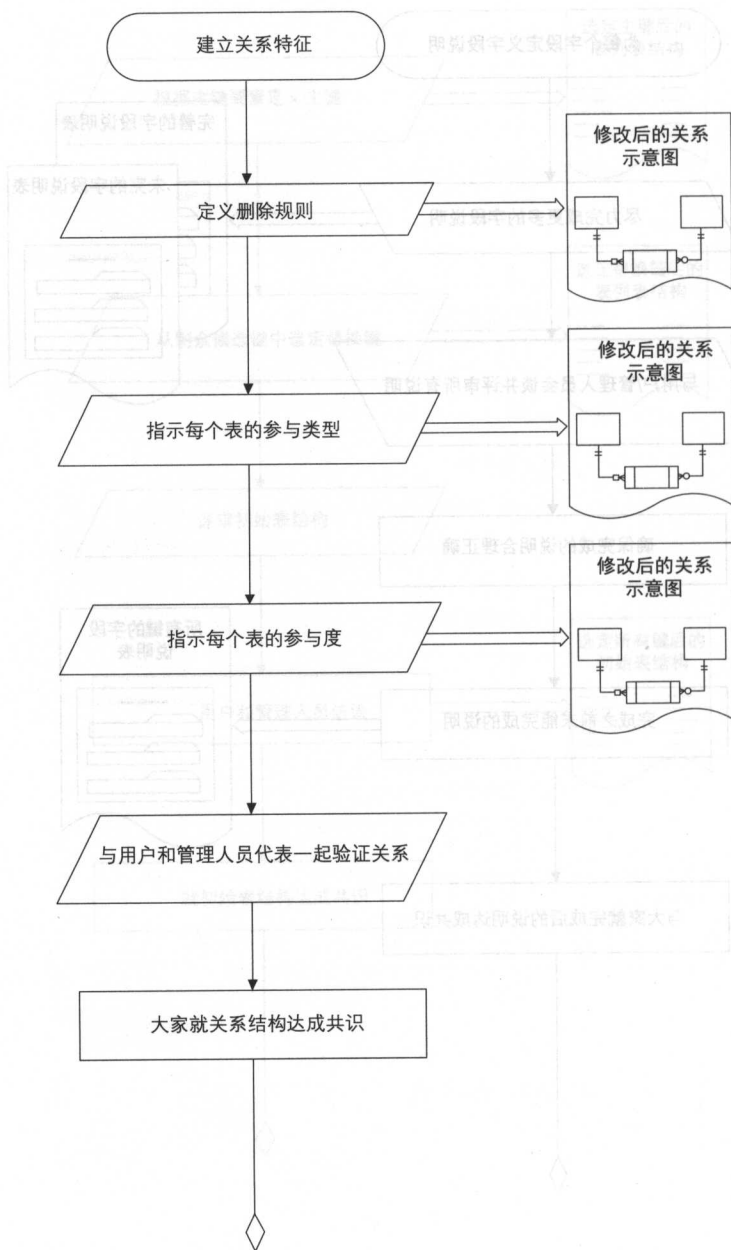


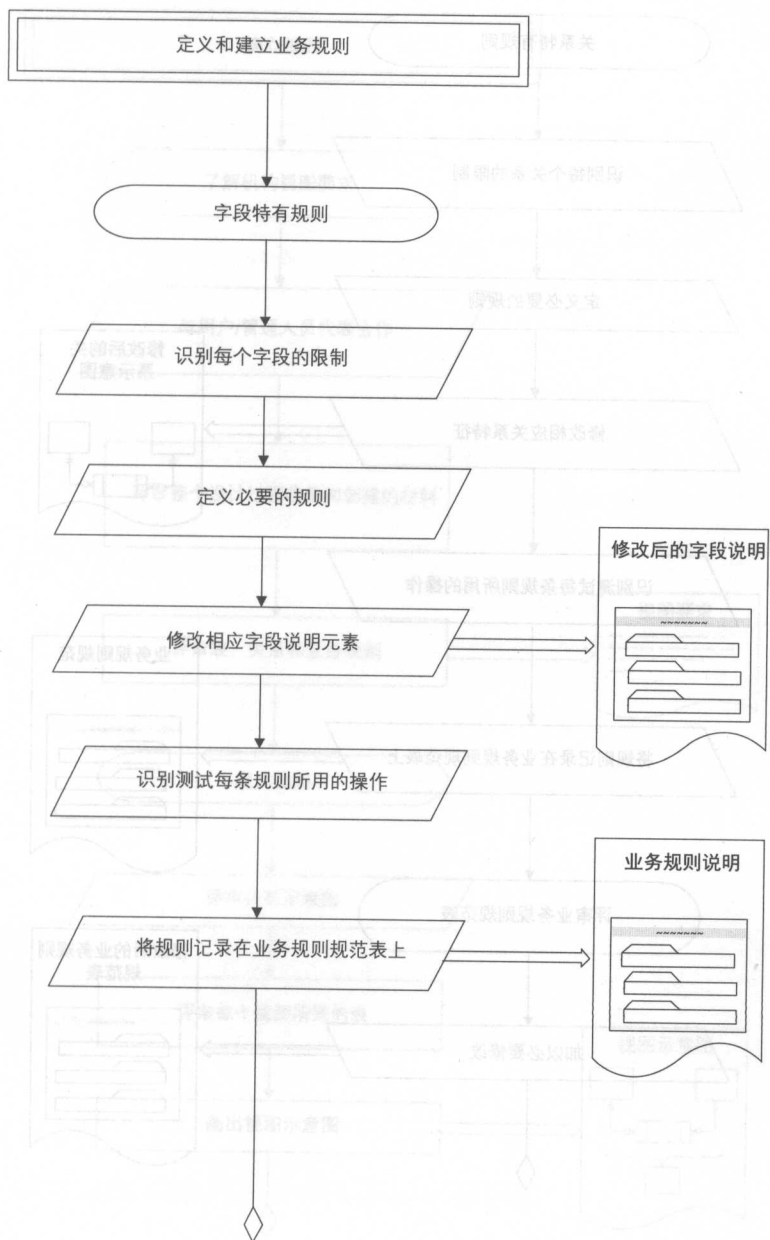


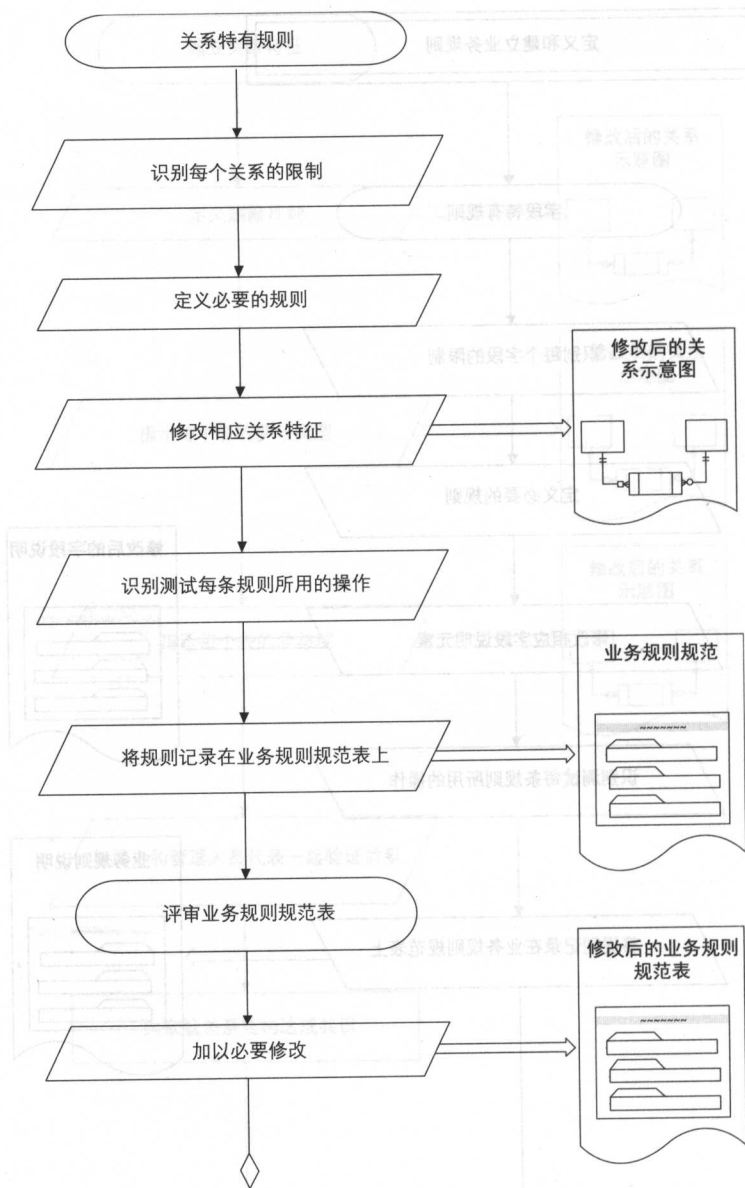


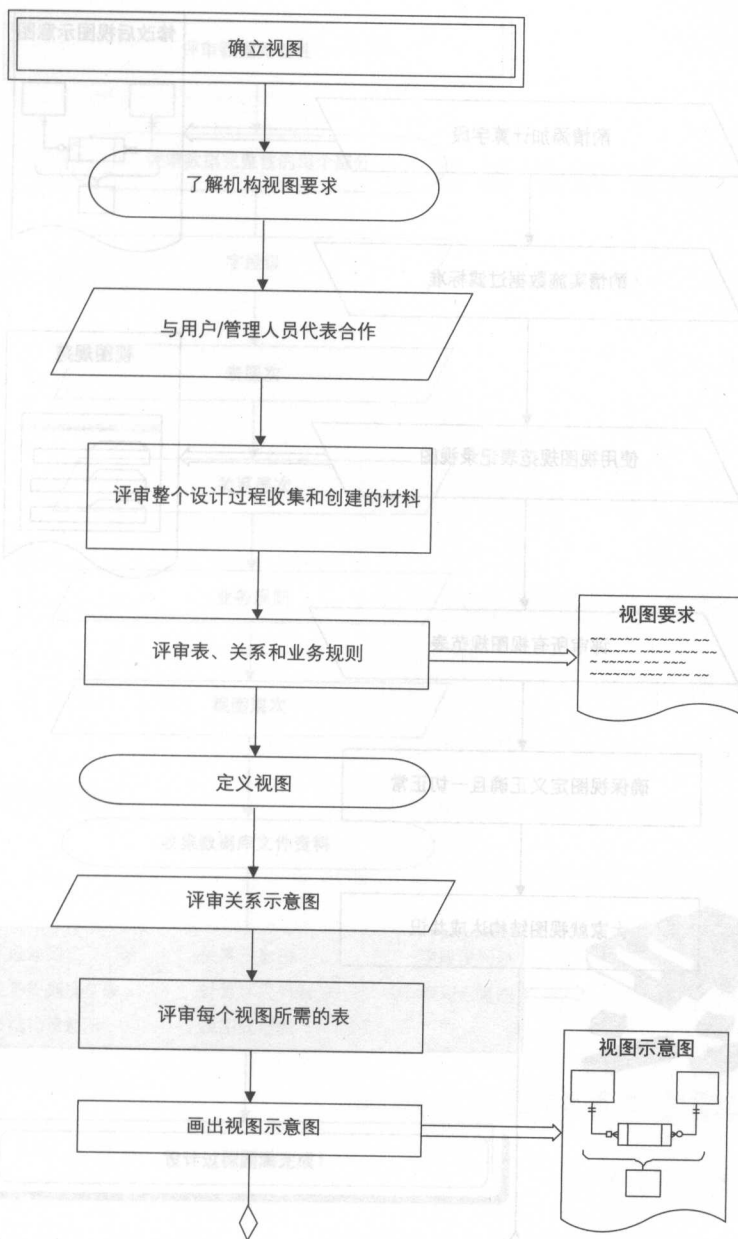


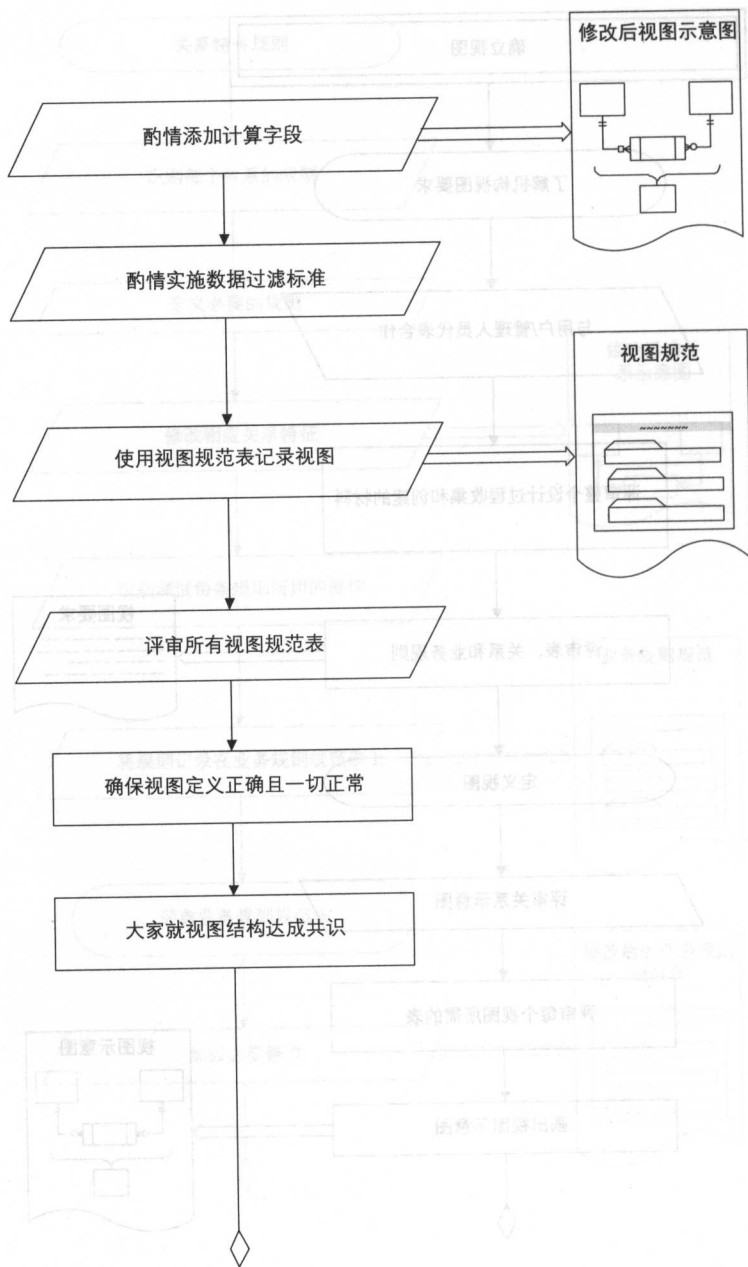


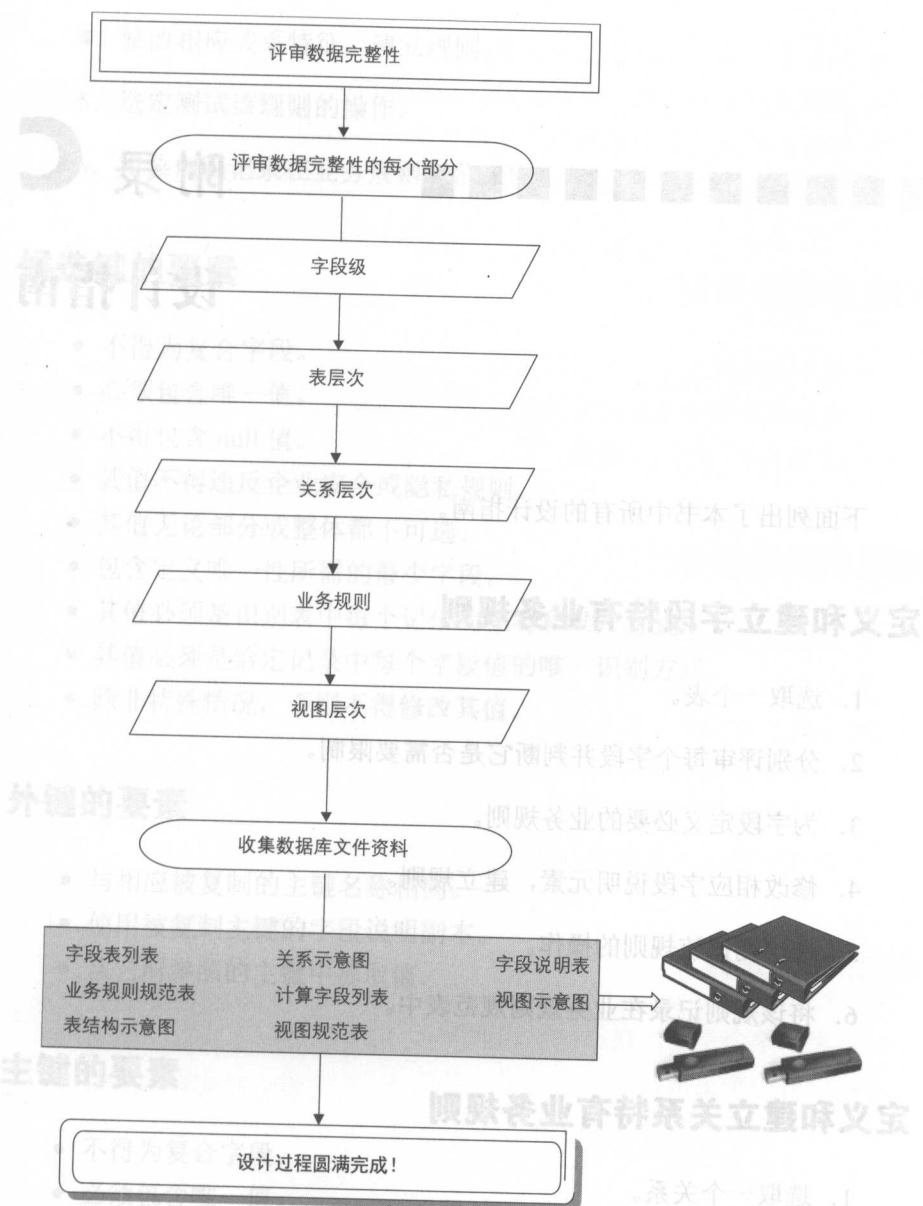














下面列出了本书中所有的设计指南。

定义和建立字段特有业务规则

1. 选取一个表。
2. 分别评审每个字段并判断它是否需要限制。
3. 为字段定义必要的业务规则。
4. 修改相应字段说明元素，建立规则。
5. 选定测试该规则的操作。
6. 将该规则记录在业务规则规范表中。

定义和建立关系特有业务规则

1. 选取一个关系。
2. 评审该关系并判断它是否需要限制。
3. 为该关系定义必要的业务规则。

4. 修改相应关系特征，建立规则。
5. 选定测试该规则的操作。
6. 将该规则记录在业务规则规范表中。

候选键的要素

- 不得为复合字段。
- 必须包含唯一值。
- 不得包含 null 值。
- 其值不得违反企业安全或隐私规则。
- 其值无论部分或整体都不可选。
- 包含定义唯一性所需的最少字段。
- 其值必须是识别表中每个记录的独特和唯一方式。
- 其值必须是给定记录中每个字段值的唯一识别方式。
- 除非特殊情况，否则不得修改其值。

外键的要素

- 与相应被复制的主键名称相同。
- 使用被复制主键的字段说明副本。
- 从它所参照的主键中提取值。

主键的要素

- 不得为复合字段。
- 必须包含唯一值。
- 不得包含 null 值。
- 其值不得违反企业安全和隐私规则。
- 其值无论部分或整体都不可选。

- 包含定义唯一性所需的最少字段。
- 其值必须是识别表中每个记录的独特和唯一方式。
- 其值必须是给定记录中每个字段值的唯一识别方式。
- 除非特殊情况，否则不得修改其值。

创建主键的规则

- 每个表有且仅有一个主键。
- 数据库中每个主键必须为唯一的，即任意两个表不得有相同的主键，除非其中一个为子集表。

理想字段的要素

- 代表表主题的鲜明特征。
- 仅包含一个值。
- 无法分解为更小的元素。
- 不含计算值或串联值。
- 在整个数据库结构中独一无二。
- 即使出现在多个表中，其主要特性始终保持不变。

理想表的要素

- 表示单个主题，包括物体或事件。
- 拥有一个主键。
- 不含复合字段或多值字段。
- 不含计算字段。
- 不含无用的重复字段。
- 冗余数据量保持最低水平。

字段层次完整性

这种类型的完整性能确保以下优势。

- 字段特性和用途明确，所有相关表都得到正确识别。
- 整个数据库中字段定义一致。
- 字段的值一致且有效。
- 修改、比较和操作字段值所运用的方法有了明确界定。

制订字段描述的指南

- 准确描述该字段，阐明其用途。
- 陈述简明扼要。
- 避免重申或改述使用字段名称。
- 避免使用专业术语、缩略语和缩写。
- 切勿包含具体实施信息。
- 不同字段描述之间保持独立。
- 切勿使用示例。

制订表描述的指南

- 包含表的准确定义。
- 解释该表对企业的重要性。
- 描述务求简明扼要。
- 避免提及具体操作信息，比如该表使用的方式和适用场合。
- 不同表描述之间保持独立。
- 避免使用示例。

创建字段名称的指南

- 字段名称应独特且富有内涵。
- 字段名称应简明扼要，准确描述字段所代表的特征。
- 字段名称应务求精简，避免拖沓冗长。
- 切勿使用缩略语，慎用缩写词。
- 切勿使用混淆字段名称含义的词语。
- 避免使用隐含或显示多个特征的名称。
- 使用名称的单数形式。

创建表名称的指南

- 表名称应独特且富有内涵。
- 表名称应明白无误地展现其主题。
- 表名称应尽量精简。
- 避免使用描述物理特征的词语。
- 避免使用缩略语。
- 避免使用专有名称或其他过多限制输入数据的词语。
- 避免使用隐含或显示多个主题的名称。
- 使用复数形式。

识别关系

根据以下步骤识别表矩阵中表间正式关系。

1. 选取两个表，注意两表交会的条目。
2. 在矩阵中查找与第一个表同一侧的第二个表，注意它和矩阵另一侧的第一个表交会的条目。

3. 对这两个条目运用相应的公式，识别这两表之间的正式关系。

$$1:1 + 1:1 = 1:1$$

$$1:N + 1:1 = 1:N$$

$$1:N + 1:N = M:N$$

4. 画出该关系的示意图。

5. 在矩阵中划掉这两个条目。

识别视图要求

根据以下步骤识别企业视图要求。

- 和代表们一道回顾你的记录。
- 回顾设计早期阶段收集的数据输入、报表和演示样本。
- 检查表及其所表示的主题。
- 分析表关系。
- 研究业务规则。

访谈指南

参与者指南

- 让参与者知晓你的意图。
- 让参与者知道你欣赏他们的参与，且他们在访谈中的回应对整个设计工程都具有一定意义。
- 如果产生争议，确保每个人都知道你是正式仲裁人。

采访者指南

- 访谈室应选取光线充足、远离嘈杂的房间，并配备大桌子和舒适的座椅。

- 每次访谈限制人数不超过 10 人。
- 对用户和管理人员分别访谈。
- 如必须对多组人进行访谈，可为每个组安排一个组长。
- 在访谈之前准备好问题。
- 如果你不擅长做笔记，就为每次访谈安排可靠的记录者或经小组许可后使用数字记录器（录音机）对访谈进行记录。
- 给予每个人同等的关注。
- 保持访谈的节奏。
- 始终控制好访谈。

宗旨

优良的宗旨应具备以下特征。

- 表述明确直接。
- 避免语言烦琐或描述过于细化，保证条理清晰。
- 避免出现具体任务的直接表述。
- 获得你（数据库开发者）和服务对象的一致认可。

任务目标

优良的任务目标应具备以下特征。

- 一般为一个陈述句，对一般任务定义简单明了，不拖泥带水。
- 多采用通用术语，语言简练、切中要点、准确明白。
- 获得你和服务对象的一致认可。

关系层次完整性

这种类型的完整性能确保以下优势。

- 关联中两表（或键字段）之间的连接合理健全。
- 以一种有意义的方式向每个表中插入新记录。
- 删除现有记录不会带来任何不利影响。
- 合理限制关系中相关记录的数量。

消除多值字段

根据以下通用步骤消除多值字段。

- 将该字段从表中移除，以之作为基础创建一个新的表。如有必要，根据之前所学的字段命名指南为其重新命名。
- 取原表的主键，将之加入新表的结构中。这个字段将在新表中发挥两个作用：它将作为该表的复合主键；它将作为外键，帮助建立新表与原表之间的关系。
- 为新表制订合适的名称、类型和描述，并将之添加到最终表列表中。

表层次完整性

这种类型的完整性能确保以下优势。

- 表中无重复记录。
- 主键为表中每个记录的唯一识别方式。
- 每个主键值是唯一的。
- 主键值不为 null。

附录 D

文档形式

下面将提供字段说明表、业务规则规范表和视图规范表的版式，供读者借鉴和使用。

本书配套资源也包含 Microsoft Word 2010 版的模板。

FIELD SPECIFICATIONS

General Elements

Field Name:	Specification Type: <input type="checkbox"/> Unique <input type="checkbox"/> Generic <input type="checkbox"/> Replica
Parent Table:	Source Specification:
Label:	
Shared By:	
Alias(es):	
Description:	

Physical Elements

Data Type:	Character Support:
Length:	<input type="checkbox"/> Letters (A-Z) <input type="checkbox"/> Keyboard (., / \$ # %)
Decimal Places:	<input type="checkbox"/> Numbers (0-9) <input type="checkbox"/> Special (© ® ™ Σ π)
Input Mask:	
Display Format:	

Logical Elements

Key Type:	<input type="checkbox"/> Non <input type="checkbox"/> Primary <input type="checkbox"/> Foreign <input type="checkbox"/> Alternate	Edit Rule:
Key Structure:	<input type="checkbox"/> Simple <input type="checkbox"/> Composite	<input type="checkbox"/> Enter Now, Edits Allowed
Uniqueness:	<input type="checkbox"/> Non-unique <input type="checkbox"/> Unique	<input type="checkbox"/> Enter Now, Edits Not Allowed
Null Support:	<input type="checkbox"/> Nulls Allowed <input type="checkbox"/> No Nulls	<input type="checkbox"/> Enter Later, Edits Allowed
Values Entered By:	<input type="checkbox"/> User <input type="checkbox"/> System	<input type="checkbox"/> Enter Later, Edits Not Allowed
Required Value:	<input type="checkbox"/> No <input type="checkbox"/> Yes	<input type="checkbox"/> Not Determined At This Time
Default Value:		
Range of Values:		
Comparisons Allowed:	<input type="checkbox"/> Same Field <input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <= <input type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <= <input type="checkbox"/> Value Expression <input type="checkbox"/> All <input type="checkbox"/> = <input type="checkbox"/> > <input type="checkbox"/> >= <input type="checkbox"/> ≠ <input type="checkbox"/> < <input type="checkbox"/> <=	
Operations Allowed:	<input type="checkbox"/> Same Field <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation <input type="checkbox"/> Other Fields <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation <input type="checkbox"/> Value Expression <input type="checkbox"/> All <input type="checkbox"/> + <input type="checkbox"/> - <input type="checkbox"/> x <input type="checkbox"/> ÷ <input type="checkbox"/> Concatenation	

BUSINESS RULE SPECIFICATIONS		
Rule Information		
Statement: _____		
Constraint: _____		
Type: <input type="checkbox"/> Database Oriented <input type="checkbox"/> Application Oriented	Category: <input type="checkbox"/> Field Specific <input type="checkbox"/> Relationship Specific	Test On: <input type="checkbox"/> Insert <input type="checkbox"/> Update <input type="checkbox"/> Delete
Structures Affected		
Field Names: _____		
Table Names: _____		
Field Elements Affected		
Physical Elements		
<input type="checkbox"/> Data Type	<input type="checkbox"/> Decimal Places	<input type="checkbox"/> Input Mask
<input type="checkbox"/> Length	<input type="checkbox"/> Character Support	<input type="checkbox"/> Display Format
Logical Elements		
<input type="checkbox"/> Key Type	<input type="checkbox"/> Values Entered By	<input type="checkbox"/> Comparisons Allowed
<input type="checkbox"/> Key Structure	<input type="checkbox"/> Required Value	<input type="checkbox"/> Operations Allowed
<input type="checkbox"/> Uniqueness	<input type="checkbox"/> Default Value	<input type="checkbox"/> Edit Rule
<input type="checkbox"/> Null Support	<input type="checkbox"/> Range of Values	
Relationship Characteristics Affected		
<input type="checkbox"/> Deletion Rule	<input type="checkbox"/> Type of Participation	<input type="checkbox"/> Degree of Participation
Action Taken		
<div style="border: 1px solid black; width: 100%; height: 100%;"></div>		

VIEW SPECIFICATIONS

General Information

Name:

Type: ☐ Data ☐ Aggregate ☐ Validation

Description:

Base Tables

Calculated Field Expressions

Field Name

Expression

Filters

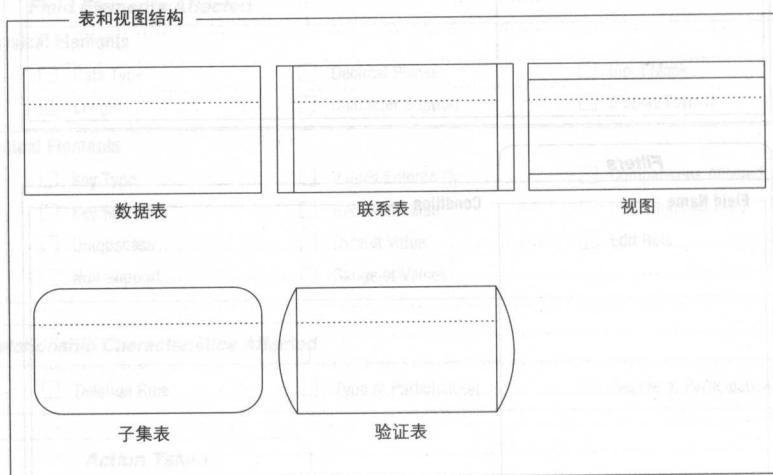
Field Name

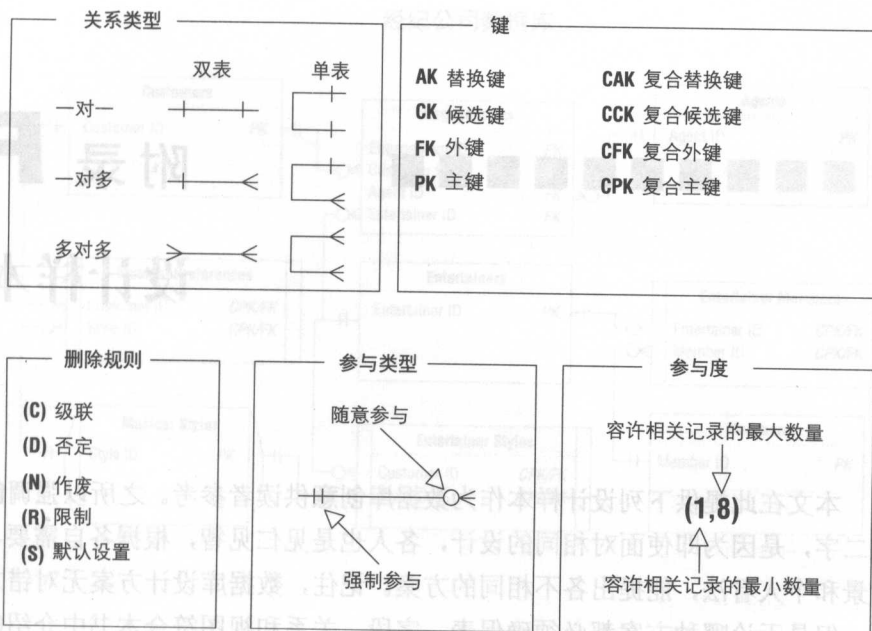
Condition

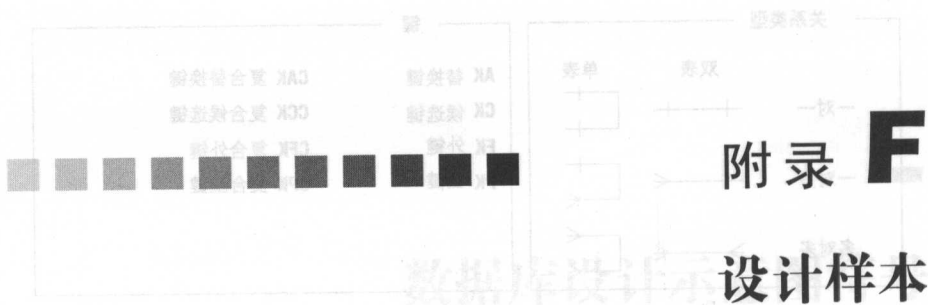
附录 E

数据库设计示意图符号

下面列举出本书中用过的与数据结构、关系、关系特征和键相关的示意图符号，以供参考。





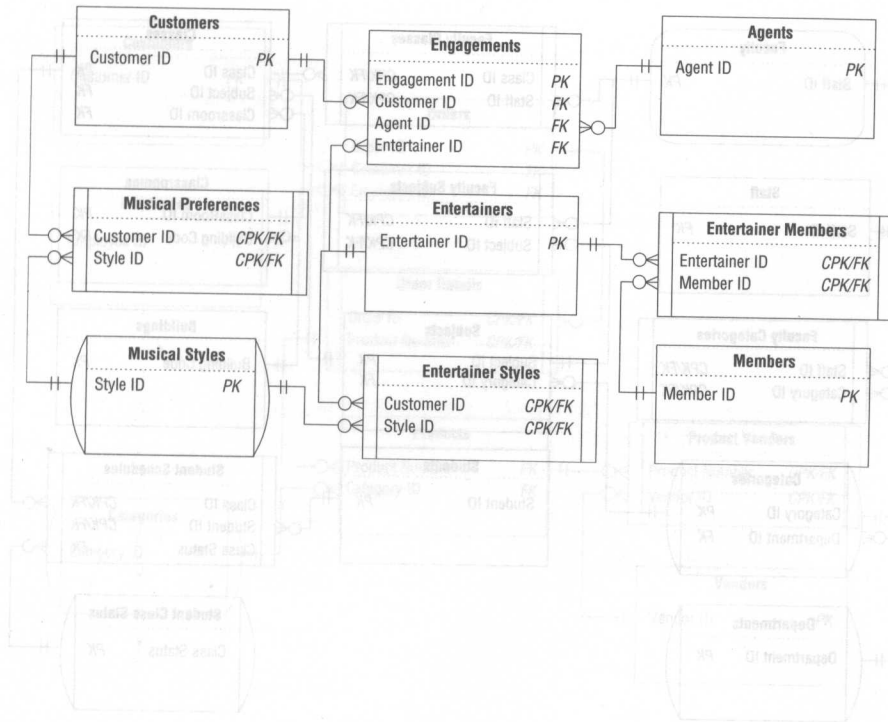


本文在此提供下列设计样本作为数据库创意供读者参考。之所以强调创意二字，是因为即使面对相同的设计，各人也是见仁见智，根据各自需要、背景和个人看法，能提出各不相同的方案。记住，数据库设计方案无对错之分，但是无论哪种方案都必须确保表、字段、关系和视图符合本书中介绍的相应指南。

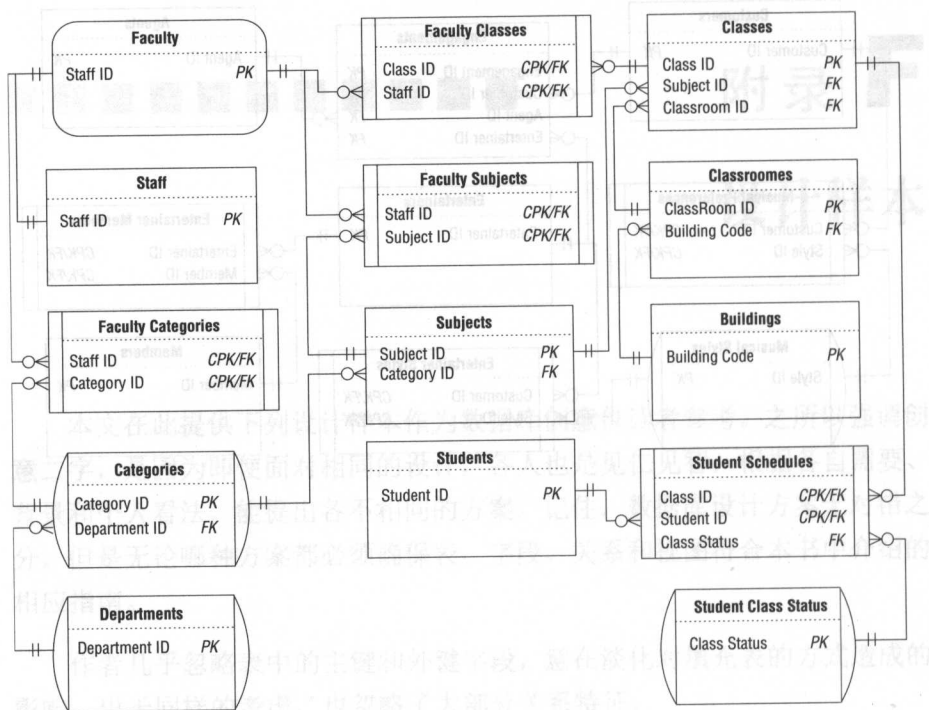
作者几乎忽略表中的主键和外键字段，意在淡化对填充表的方式造成的影响。出于同样的考虑，也忽略了大部分关系特征。

希望读者能从下列设计中找到能够运用的方法，将它当作真实数据库，模拟整个数据库设计过程。在过程结束之时，相信你已有适合自己需要的数据库。

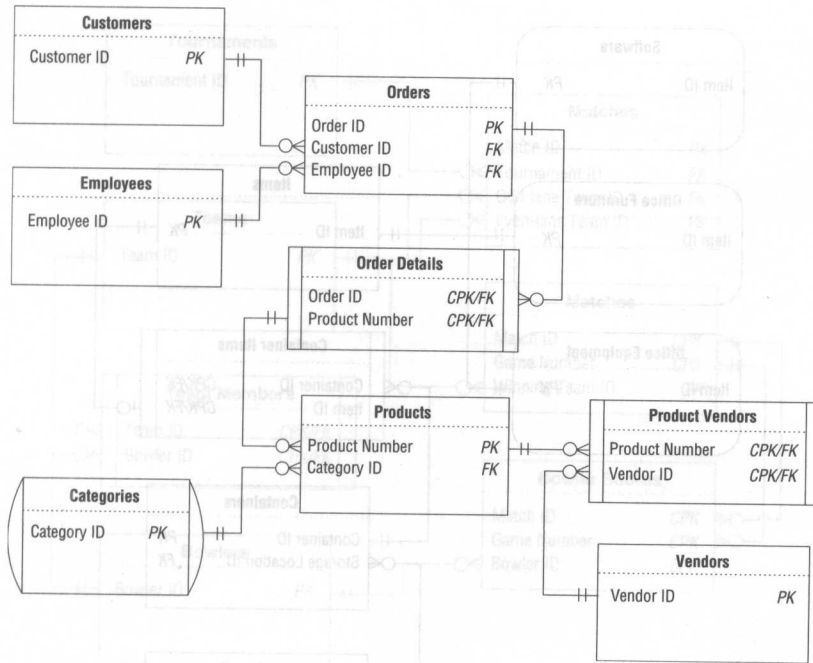
经纪公司数据库



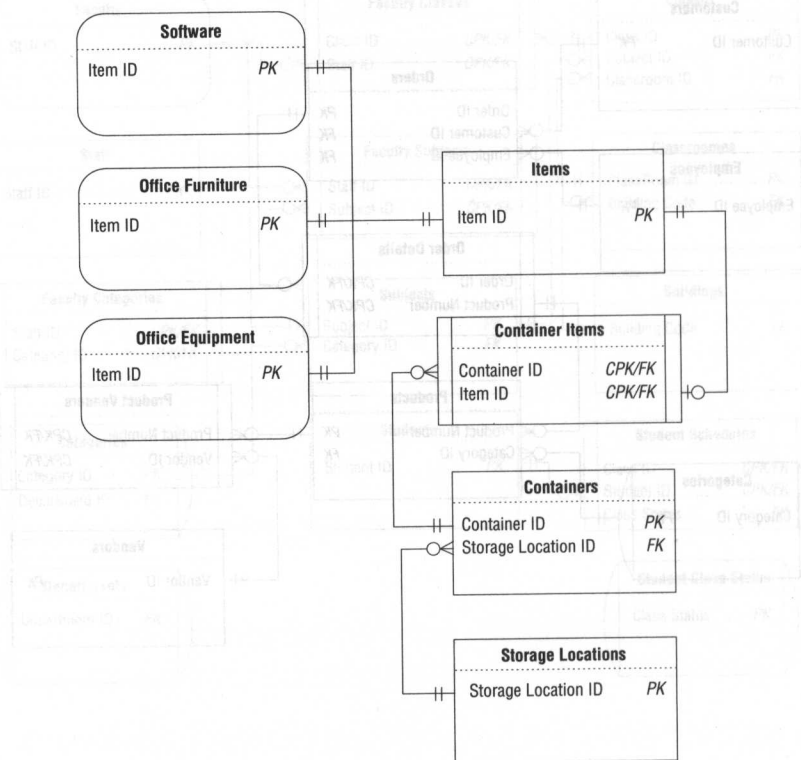
学校数据库



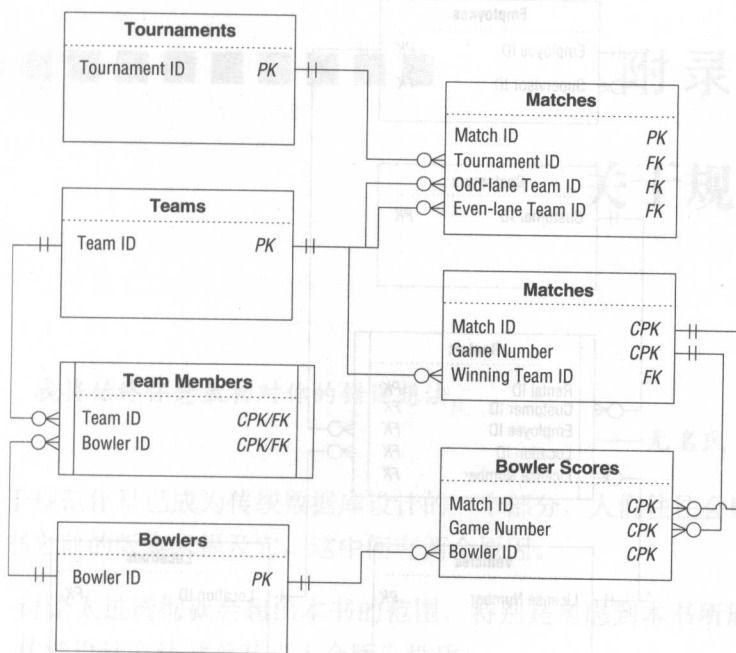
销售订单数据库



办公用品库存数据库



保龄球联赛数据库

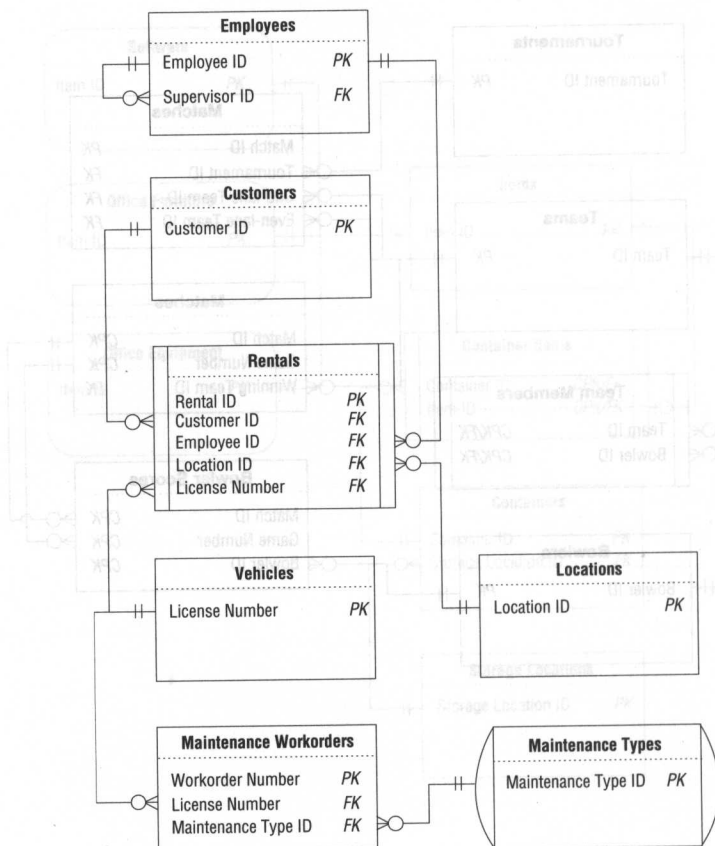


请注意

在此之前，需要明确以下几点。

- 阅读本附录之前，请先阅读第2章末的“本书中展示的设计方法”和“规范化”部分。这两个部分对读者为什么以及如何提出其设计方法

车辆租赁数据库





附录 G

关于规范化

我将始终怀念最初对你的错误想法。

——无名氏

鉴于规范化早已成为传统数据库设计的一个部分，人们往往会疑惑，为什么本书之前的版本未提及它。这中间有两个原因。

1. 讨论太过透彻就会超出本书的范围，特别是考虑到本书所展示的非传统设计方法学及其“大众版”性质。
2. 规范化实际已融入到本书的设计过程中。（下文解释。）

本书亚马逊网站的页面上，仍然有网友对此提出问题和发表评论，所以下文将讨论本书所介绍的设计过程是如何融合传统规范化过程的。之后，想必当前学习传统设计和规范化过程的读者能够更为清楚地理解本书中的设计方法学。

请注意

在此之前，需要明确以下几点。

- 阅读本附录之前，请先阅读第2章末的“本书中展示的设计方法”和“规范化”部分。这两个部分对作者为什么以及如何提出其设计方法

学做了完整的说明，也将为更好地理解下文提供一定背景知识。

- 这并非对传统规范化过程的正式讨论或指导。读者如有兴趣可参见附录 H “推荐书目”，其中包含的一些书籍深刻而透彻地讨论了此话题。
- 此处以读者早已理解传统规范化过程及其相关概念和术语为前提。笔者发现，通常对这一讨论感兴趣的群体是早期数据库应用程序员，他们对规范化非常熟悉，还有就是学习规范化的学生。有鉴于此，笔者专为这两个群体定制了本附录。
- 切勿将规范化与本书中的设计方法学逐字逐句一一映射。规范化确实融入了本书的方法学中，但是并非全套照搬。规范化是传统逻辑设计过程中的一个特定环节，本书方法学将之巧妙融入了整个设计过程。随着阅读的深入，这将变得越加明晰。
- 根据本书中的设计方法学，能制定出十分规范的表。不过，无论是传统方法学还是本书中的方法学，都必须严格遵循。走捷径或偷工减料，必将在表结构和完整性方面埋下隐患。

但愿，读者刚刚读完了第 2 章的两节。本书接下来将解释作者的设计方法学如何融合规范化。

简要回顾

现在，本文将简要回顾笔者如何提出自己的方法学，这在第 2 章的两节已有介绍。

不过，在此之前先来看看传统设计方法的整套流程。

- 识别主要实体
- 识别关系类型
- 确定主键
- 确定外键
- 联系属性与视图或关系类型
- 确定属性域
- 运用规范化验证模型

- 定义完整性限制

当初，这种方法学困扰我最多的两个地方，是规范化过程（整体）和似乎无穷无尽的迭代。

我早就明白，规范化的目的是将设计不当的表转化为结构健全的表。我也熟知这个过程：取给定的表，对照范式进行测试，判断它的设计是否规范。如果不符合规范，就做出适宜的修改，再测试，并不断重复，直到该表结构达到完善。图 G.1 所示为这个过程的示意图。

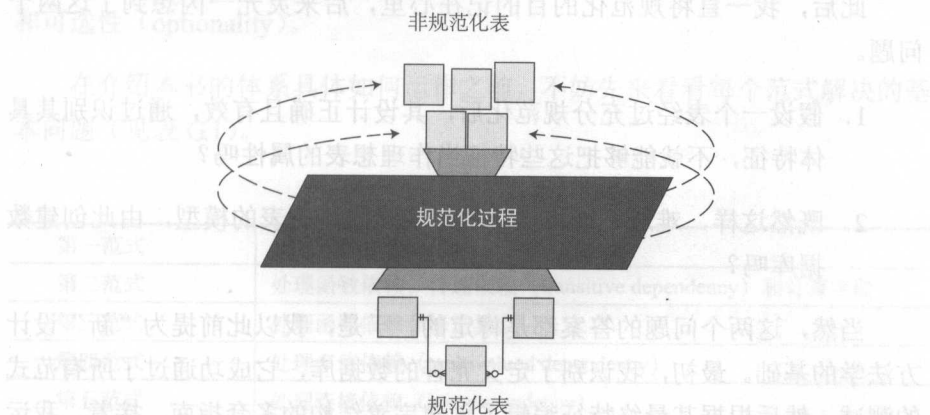
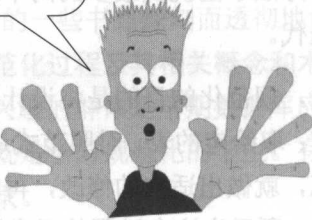


图 G.1 一般规范化过程示意图

范式有多种，每种范式都用于测试特定类的问题和特征，比如修改异常、函数依赖、传递依赖、多值依赖、连接依赖、域和键。如果没有研究过正式关系数据库理论，就会对范式感觉困惑不解。

我一度自问，“为什么我们在花去 3/4 的时间创建数据库后，戛然而止，转而判断数据库的结构是否规范？”我觉得这种方法实在可笑。



等等！我们是否拥有完善的结构呢？

此后，我一直将规范化的目的记在心里，后来灵光一闪想到了这两个问题。

1. 假设一个表经过充分规范化后，其设计正确且有效，通过识别其具体特征，不就能够把这些特征当作理想表的属性吗？
2. 既然如此，难道不能将这个理想表作为所有表的模型，由此创建数据库吗？

当然，这两个问题的答案都是肯定的。于是，我以此前提为“新”设计方法学的基础。最初，我识别了定义完善的数据库，它成功通过了所有范式的测试，然后根据其最终特征编辑了创建完善结构的多套指南。接着，我运用这些指南创建了新数据库的表结构，并修正了一个现有数据库的表结构缺陷，然后对这两个数据库的表结构开展了测试。这些测试反映的情况都非常好，所以我决定对整套传统设计方法学运用这项技巧。随后，我制订的指南解决了传统设计方法出现的其他问题，比如域、子类型、关系、数据完整性和参照完整性。待新指南全部完成后，我又执行了更多的测试，发现我的方法学非常有效。

我的设计方法学省去了传统设计方法学的多个方面，而这些往往是数据库开发新手望而生畏的。例如，传统意义的规范化变得更为清楚，因为它（借助指南）融入了整个设计过程。另外，我的方法学条例清晰、易于执行，我认为这是由于指南直白易懂，大多数人都能理解。

个人认为，设计数据库的过程不是也不应难以理解。只要将过程直白地

展示出来，将每个概念或技巧解释清楚，任何人都能设计规范数据库。

规范化如何融入本书的设计方法学

如前所述，规范化和本书中的设计方法学并非直接一一映射。本书的方法学通过直观的手段解决之前由规范化解决的问题。另外，还能解决其他的传统设计问题，比如标量值（scalar value）、确定问题（determinate）、函数依赖（functional dependency）、域、修正异常、参照完整性、基数（cardinality）和可选性（optionality）。

在介绍本书的体系具体如何运作之前，不妨先来看看每个范式解决的基本问题（见表 G.1）。

表 G.1 范式解决的基本问题

第一范式	处理函数和多值依赖
第二范式	处理函数依赖、传递依赖（transitive dependency）和计算字段
第三范式	处理函数依赖和修正异常
第四范式	处理多值依赖（multivalued dependency）
第五范式	处理连接依赖（join dependency）
第六范式	主要用于空间数据（spatial data）
Boyce-Codd范式	处理确定问题和候选键
域/键范式	处理域和键

我将这些范式（以及本书先前引用的其他问题）记在心里，最初想要将它们融入新的设计方法学中，使之变得更为有效且更为精简。同时，我想让新的方法学清楚易懂。这正是我决定摆脱传统术语和数学方法，而使用直白的语言进行描述的原因。

表 G.2 所示为本书设计方法解决传统规范化和设计问题的情况。

表 G.2 本书设计方法解决传统规范化和设计所面对的问题的情况

本书设计方法学组成部分	传统规范化或设计解决的问题
业务规范	逻辑域, 验证表
候选键/主键的要素	函数依赖、多值依赖、传递依赖、确定问题
外键的要素	外键、参照完整性、修正异常
理想字段的要素	标量值、多值字段和复合字段、计算值
理想表的要素	函数依赖、多值依赖、传递依赖、连接依赖、重复字段、重复数据和冗余数据、修正异常、子类型
字段级完整性	标量值、物理域、逻辑域、域完整性
字段说明	标量值、物理域、逻辑域、域完整性
关系特征	基数、可选性、删除规则
关系层次完整性	参照完整性、外键
消除复合字段	标量值、逻辑域
消除多值字段	标量值、逻辑域
表层次完整性	主键完整性、重复记录、函数依赖

逻辑设计与物理设计及实现

有时, 我会被问及为什么没有详细讨论 SQL 和实现问题, 比如索引、划分和分布。答案十分简单: 我始终认为逻辑设计过程应与物理设计和实现过程分开。

我也认为, 许多人都在无意之间掉进了单凭用于实现的 RDBMS 软件设计数据库的陷阱。多数情况下, 他们这样做的原因是对特定 RDBMS 具有一定的熟练度, 或者他们的机构或组织早已使用了特定 RDBMS。这种做法并不明智, 读者应引以为戒, 原因如下。

- 可能基于主观判断对设计做出决策。例如, 你也许决定不对给定关系施加参与度, 因为你认为该 RDBMS 不提供施加参与度的方法。
- 由于疏忽大意, 让 RDBMS 指定的数据库设计完全背离了机构信息要求。这种情况通常发生在发现 RDBMS 对数据库的特定方面(比如, 字段说明和关系特征)只提供有限支持的时候。
- 设计受自身的 RDBMS 知识所限。例如, 也许决定不执行关系特征,

原因仅仅是不知道其具体做法。

- 设计受自身对 RDBMS 的熟练程度所限。熟练程度影响了实现数据库的各个方面的效率和有效程度，比如字段说明和业务规则。
- 使用这种方法设计数据库通常导致结构设计不当，数据完整性不足，以及数据不一致和信息不准确。在 RDBMS 中定义数据库貌似轻松。这样创建的数据库也许能够运作，但是极有可能埋下隐患。
- 最后，熟知且钟爱的 RDBMS 也许不能适应机构数据库要求。

我个人看来，读者应该始终在不考虑任何 RDBMS 的前提下，设计数据库的逻辑结构。这样，才更有可能设计出结构完善的数据库，因为可以专注于机构信息要求。设计完成后，就可以清楚地判断如何实现数据库（单用户应用程序、客户端/服务器、基于网络等）以及使用哪种 RDBMS 便于实现。

我希望这最终能澄清和解答任何与本书设计方法学和规范化相关的困惑和问题。读者如能通过阅读本附录，对我的方法有更进一步的了解，并明白它与规范化解决相同问题的原理，我的目的也就达到了。

附录 H 推荐书目

附录 H

推荐书目

如果读者希望深入学习数据库技术，以下是我推荐阅读的书籍。笔者选定的这些书籍都已经历过时间的考验，成为了数据库产业和相关学术机构的“标准读物”。（笔者的作品也有幸名列其中。）不过，需要注意其中大多数书籍都具有一定挑战性。笔者推荐这些书籍的前提是读者拥有计算和编程方面丰富的背景知识，或者正在攻读计算机科学领域的学位。

Codd, E. F. (1990) *The Relational Model for Database Management: Version 2*. Reading, MA: Addison-Wesley. ISBN: 0201141922 （注意：这本书虽然很难找到，但是对于真正的数据库开发者，绝对值得收藏。由“关系数据库之父”本人编著。）

Connolly, Thomas, and Carolyn Begg. (2009) *Database Systems—A Practical Approach to Design, Implementation, and Management, Fifth Edition*. Boston: Addison-Wesley. ISBN: 0321523067

Date, C. J. (2001) *The Database Relational Model—A Retrospective Review and Analysis*. Boston: Addison-Wesley. ISBN: 0201612941

——(2003) *An Introduction to Database Systems, Eighth Edition*. Boston: Addison-Wesley. ISBN: 0321197844

——(2006) *Databases, Types and the Relational Model, Third Edition*.

Boston: Addison-Wesley. ISBN: 0321399420

——(2005) *Database In Depth—Relational Theory for Practitioners*. Sebastopol, CA: O'Reilly Media, ISBN: 0596100124

——(2012) *Database Design and Relational Theory: Normal Forms and All That Jazz*. Sebastopol, CA: O'Reilly Media. ISBN: 1449328016

Hoffer, Jefferey A., Mary B. Prescott, and Fred R. McFadden. (2010) *Modern Database Management, Tenth Edition*. Upper Saddle River, NJ: Prentice Hall. ISBN: 0136088392

Kroenke, David M. (2011) *Database Processing, Twelfth Edition*. Upper Saddle River, NJ: Prentice Hall. ISBN: 0132145375

术 语 表



即席信息检索 (Ad Hoc Information Retrieval) 运用即席查询检索信息的过程, 检索的信息为当前现有报表或数据管理界面未出现的信息。

即席查询 (Ad Hoc Query) 向数据库应用程序发出的非预定义查询。

聚合函数 (Aggregate Function) 程序编码的片段, 对数据集执行特定类型的数学聚合并返回一个值。

聚合视图 (Aggregate View) 一种视图, 用于显示聚合特定数据集所产生的信息。

替换键 (Alternate Key) 未被指定为主键的候选键。

分析型数据库 (Analytical Database) 一种存储静态数据的数据库。用于追踪趋势, 观察长时间段内的统计数据, 以及制订战略战术业务预测; 通常与 OLAP 相联系。

应用 (Application) 一种商业或定制软件程序, 通常用于为数据库提供用户友好界面。

应用开发 (Application Development) 设计和创建应用程序的过程, 完成的应用程序将作为数据库的用户界面。

面向应用业务规则 (Application-Oriented Business Rule) 一种规则, 所施加的限制可以在数据库物理设计或数据库应用程序设计过程中建立。

应用程序 (Application Program) 作为数据库用户界面的商业或定制软件。

人造候选键 (Artificial Candidate Key) 一种字段, 专为充当候选键而创建。它的存在是由于表中缺乏“自然出现”的候选键。

关联表 (Associative Table) 见联系表。

属性 (Attribute) 在关系模型中等同于字段。

基表 (Base Table) 作为视图基础的表。

业务规则规范 (Business Rule Specification) 表示一条业务规则的所有特征, 比如规则语句、所施加的限制、影响的结构等。

业务规则 (Business Rule) 根据企业认识和使用其数据的方式, 对数据库的特定方面施加的限制。

计算字段 (Calculated Field) 一种字段, 包含级联文本值或数学表达式结果。

计算字段列表 (Calculated Field List) 一种字段列表, 只能在 RDBMS 中进行定义。(注意, 在表结构中无法定义计算字段。)

基数 (Cardinality) 关系数据库中两表之间存在的关系类型。见关系。

子表 (Child Table) 在给定关系中, 一个表包含的记录明确依赖于相关表中记录的存在, 则该表为相关表的子表。

客户端/服务器 RDBMS (Client/Server RDBMS) 一种 RDBMS, 其中的数据存放在作为数据库服务器的计算机中, 用户通过自己计算机上的应用程序与之交互, 而用户计算机上的应用程序则作为数据库客户端。

封闭式问题 (Closed Question) 其明显特征就是答案明确且数量有限。这类问题给后续问题留下的拓展空间很小。

复合主键 (Composite Primary Key) 由多个字段组成的主键。

数据 (Data) 数据库中存储的值。

数据一致性 (Data Consistency) 在整个数据库中，同一字段的值始终保持不变。

数据输入表单 (Data Entry Form) 应用程序中用于汇聚和收集数据的界面。

数据完整性 (Data Integrity) 控制数据库中数据有效性、一致性和准确性的一套规则或指南。数据完整性分为四种：表层次、字段层次、关系层次和业务规则。

数据结构 (Data Structure) 用于存储数据（比如，字段或表）的特定构造。

数据表 (Data Table) 一种表，存储用于提供信息的数据；关系数据库中最为常见的表。

数据视图 (Data View) 一种视图，用于检查和操作来自一个或多个基表中数据。

数据仓库 (Data Warehouse) 出于询问和分析目的设计的关系数据库，不能用于事务处理。

数据库应用程序 (Database Application Program) 见应用程序。

数据库设计过程 (Database Design Process) 设计数据库逻辑结构所需的整套操作。

面向数据库业务规则 (Database-Oriented Business Rule) 一种规则，所施加的限制可以在数据库逻辑设计过程中建立。

DBMS (Database Management System, 数据库管理系统) 一款用于创建、维护、修改和操作一个数据库的软件程序。

参与度 (Degree of Participation) 鉴于关系数据库中两表之间存在关系，它表示与一表中单一记录相关的另一表中记录的最大数量和最小数量。

删除规则 (Deletion Rule) 当用户发出请求删除关系中父表的给定记录, RDBMS 所应做出的反应由删除规则决定。

域 (Domain) 见字段说明。

域完整性 (Domain Integrity) 见字段层次完整性。

重复数据 (Duplicate Data) 在数据库的多个表中出现的非主键值。

重复字段 (Duplicate Field) 出现在多个表中的字段, 原因有这些: 用于将一组表联系起来; 指示特定类型的值的多次出现; 倾向于需要补充信息。

动态数据 (Dynamic Data) 不断变化且始终反应即时信息的数据。

候选键的要素 (Elements of a Candidate Key) 一套用于判断给定字段是否符合候选键条件的指南。

外键的要素 (Elements of a Foreign Key) 一套用于判断给定字段是否符合外键条件的指南。

主键的要素 (Elements of a Primary Key) 一套用于判断给定字段是否符合主键条件的指南。

理想字段的要素 (Elements of the Ideal Field) 一套指南, 用于创建完善的字段结构且能有效识别设计欠佳的字段。

理想表的要素 (Elements of the Ideal Table) 一套指南, 用于创建完善的表结构且能有效识别设计欠佳的表。

终端用户 (End User) 数据库或数据库应用程序的使用者。

终端用户应用 (End-User Application) 作为数据库用户界面的商业或定制软件。

实体完整性 (Entity Integrity) 见表层次完整性。

事件 (Event) 发生在指定时间点的某个事件 (比如, 医疗预约或股票交易), 可以用一个表进行表示。

明示信息 (Explicit Information) 在给定问题的回答中明确表述的信息。

扩展数据类型 (Extended Data Types) 许多 RDBMS 程序提供的额外数据类型, 超过了 SQL 标准指定的范围。

字段 (Field) 数据库中最小的结构, 表示所属表主题的一个特征, 也是数据库中唯一实际存储数据的结构。

字段层次完整性 (Field-Level Integrity) 这种数据完整性确保了以下几点优势: 字段特性和用途明确, 所有出现该字段的表都得到正确识别; 字段定义始终保持一致; 字段的值一致且有效; 对字段的值可运用的修改、比较和运算得到确认。

字段说明 (Field Specification) 表示一个字段的所有一般、物理和逻辑元素。(传统叫法为域。)

字段特有业务规则 (Field-Specific Business Rule) 对给定字段的字段说明元素施加限制的规则。

过滤器 (Filter) 对视图施加的一套限制, 包含一个或多个限制, 从而使该视图返回特定一组信息。

最终表列表 (Final Table List) 这个列表包含数据库中每个表的键信息 (名称、类型和描述)。

一阶谓词逻辑 (First-Order Predicate Logic) 关系模型基于的两个数学分支之一。

层次数据库 (Hierarchical Database) 这种数据库中数据的结构采用分层式, 通常可图解为一棵倒置树。

实现过程 (Implementation Process) 设计逻辑数据库和将它融入特定 RDBMS 所需的整套操作。

暗含信息 (Implicit Information) 给定问题的回答中未明示的信息; 必须通过揣摩从中挖掘出来。

索引 (Index) RDBMS 程序中的一个结构, 可用于提升数据处理。

信息 (Information) 经处理的数据, 处理的目的是发掘其中的意义, 让使用和查看该数据的人们能加以利用。

信息要求 (Information Requirement) 为了企业运作正常、有效且高效, 信息必须获得数据库中的数据的支撑。

继承数据库 (Inherited Database) 见遗留数据库。

键 (Key) 在表中发挥特定作用的特殊字段; 键的类型决定了相应表的用途。键的类型主要有四种: 候选键、主键、替换键和外键。

LAN 见局域网。

遗留数据库 (Legacy Database) 存在和使用多年或更久的数据库。

联系表 (Linking Table) 帮助建立给定两表之间多对多关联的表。

特征列表 (List of Characteristics) 一个名词的集合, 用于指示主题列表上的项的各种属性。

主题列表 (List of Subjects) 一个名词的集合, 表示企业可能感兴趣的主体。

局域网 (Local Area Network, LAN) 在相对有限的独立区域内, 共享服务和资源的一组计算机及外围设备。

逻辑子关系 (Logical Child Relationship) 两层次数据库中两表之间存在的一种关系。

逻辑数据独立性 (Logical Data Independence) 更改数据库逻辑设计不会对建立在该数据库基础上的应用程序造成不利影响。

查找表 (Lookup Table) 见验证表。

大型计算机 (Mainframe Computer) 大型、高端且极其强大的计算机, 设计目的是同时处理数以百万计的高度密集型运算。

多对多关系 (Many-to-Many Relationship) 关系数据库中两表之间存在的一种关系, 其中一表的单一记录可与另一表的多个记录相联系且反之亦然。

成员 (Member) 网状数据库中给定关系的从属节点。

缺失值 (Missing Value) 由于人为错误, 未被输入给定字段的数据值。

任务目标 (Mission Objective) 描述用户利用数据库中数据可执行的一般任务的语句。

宗旨 (Mission Statement) 确立数据库的用途并为设计工作指明重心所在。

多层次完整性 (Multilevel Integrity) 包含了多个成分: 字段层次完整性、表层次完整性、关系层次完整性或业务规则。

复合字段 (Multipart Field) 包含多个类型的值的字段。

多值字段 (Multivalued Field) 包含多个相同类型的值的字段。

网状数据库 (Network Database) 这种数据库中数据的结构采用分层式, 通常可图解为一棵倒置树。不过, 与层次数据库不同, 网状数据库可包含多棵共享分支的倒置树。

节点 (Node) 网状数据库中给定记录的集合。

非键 (Non-key) 不充当候选键、主键、替换键或外键的字段。

范式 (Normal Form) 一组特定规则, 可用于测试表结构, 确保该表完善、无问题。

规范化 (Normalization) 将大表分解为较小的表的过程, 目的是消除冗余数据和重复数据。

Null 表示缺失或未知的值; 它不表示零或一串空白字符。

对象 (Object) 一个实体 (比如一个人、地方或事物), 可表示为一个表。

OLAP (Online Analytical Processing, 联机分析处理) 显示分析型数据库中数据的一种方法, 这些数据以一个表或立方体的形式得到总结和呈现。

OLTP (Online Transaction Processing, 联机事务处理) 数据库管理系统中的一个系统, 负责计算机接收事务后即行处理, 并实时更新主文件。

一对多关系 (One-to-Many Relationship) 关系数据库中两表之间存在的一种关系, 其中一表的单一记录可与另一表的多个记录相联系, 而另一表的单一记录仅与前一表的单一记录相联系。

一对一关系 (One-to-One Relationship) 关系数据库中两表之间存在的一种关系, 其中一表的单一记录仅与另一表的单一记录相联系且反之亦然。

联机分析处理 (Online Analytical Processing) 见 OLAP。

联机事务处理 (Online Transaction Processing) 见 OLTP。

开放式问题 (Open-Ended Question) 这种问题具有两个特征: 回答方式多种多样; 易于引出进一步的问题。

操作系统 (Operating System) 一套完整的软件, 为计算机硬件、外围设备 (比如, 打印机和扫描仪) 以及所有其他软件程序管理和提供服务。缺少操作系统, 计算机便不能运作。

操作型数据库 (Operational Database) 一种存储动态数据的数据库, 用于满足收集、修改和维护日常数据的需要; 通常与 OLTP 相关。

孤立记录 (Orphaned Record) 给定两个相关的表, 孤立记录是指存在于一表中、与另一表的任一记录都不相关的记录。

系主 (Owner) 网状数据库中给定关系的主节点。

系主/成员关系 (Owner/Member Relationship) 网状数据库中的一种关系, 其中系主表可与一个或多个成员表相联系, 但是单一成员表必须与特定系主表相联系。

纸质数据库 (Paper-Based Database) 包含各种形式的资料, 索引卡、

文件夹等，用于收集和维护数据。

父/子关系 (Parent/Child Relationship) 层次数据库中的一种关系，其中父表可与一个或多个子表相联系，但单一子表只能与一个父表相联系。

父表 (Parent Table) 在给定关系中，一表中包含的记录不依赖于相关表中记录的存在，该表即为父表。

解析 (Parse) 将给定数据值分解为多个较小的部分。

物理数据独立性 (Physical Data Independence) 数据库软件供应商更改数据库物理实现，不会给建立在该数据库基础上的应用程序带来不利影响。

指针 (Pointer) 层次数据库中，形成父表与子表之间明确连接的一种机制。

初始字段列表 (Preliminary Field List) 一个表达企业基本数据要求的字段列表，包含了数据库必须定义的核心字段。

初始表列表 (Preliminary Table List) 数据库中必须定义的核心表的集合。

主键 (Primary Key) 独立识别表中每个记录的字段或字段组。

编程环境 (Programming Environment) 给定计算平台 (PC、客户端/服务器、大型计算机等)、操作系统和编程语言的组合。

编程语言 (Programming Language) 一种软件程序，可用于定义指令集，最终这些指令集将被计算机处理和执行。

查询 (Query) 通过 SQL 查询语句向数据库发出的一个信息请求。

查询生成器 (Query Builder) 数据库软件程序中的一个工具，让用户能通过易操作的图形界面创建查询。

RDBMS (Relational Database Management System, 关系数据库管理系

统) 一种软件系统, 用于创建、维护、修改和操作关系数据库。

记录 (Record) 表中的一个结构, 由表中每个字段的单一值组成, 表示该表主题的一个唯一的实例。

递归关系 (Recursive Relationship) 见自引用关系。

冗余数据 (Redundant Data) 在一个字段中重复出现的一个值, 成因是该字段参与建立两表之间的关联或某字段或表出现异常。

参考字段 (Reference Field) 见重复字段。

参照完整性 (Referential Integrity) 见关系层次完整性。

关系 (Relation) 在关系模型中, 等同于表。

关系数据库 (Relational Database) 数据库的一种, 将数据以关联的形式存储 (用户将之视为表)。每个关联由元组 (记录) 和属性 (字段) 组成。

关系数据库管理系统 (Relational Database Management System) 见 RDBMS。

关系模型 (Relational Model) 埃德加·科德博士基于集合论和一阶谓词逻辑建立的数据模型。

关系 (Relationship) 两表之间存在的依赖关系, 前提是一表的记录可与另一表的记录产生联系。关系数据库中存在三种关系: 一对一、一对多和多对多。

关系示意图 (Relationship Diagram) 给定两表之间或同一表中给定记录集之间的关系示意图。

关系层次完整性 (Relationship-Level Integrity) 数据完整性的一种, 确保表之间关系完善和这些表中记录始终同步, 无论在其中输入、更新或删除数据。

网页 (Web Page) 一个文件, 由一个超文本标记语言 (HTML) 文件

关系特有业务规则 (Relationship-Specific Business Rule) 这种规则施加的限制对关系的特征造成影响。

报表 (Report) 任何手写、打印或计算机生成的文档，通过整理和显示数据以便人们能获得有用信息。

根表 (Root Table) 层次数据库结构中最顶层的表。

屏幕演示 (Screen Presentation) 一系列的界面，用于有序地探讨各种主题。

多对多自引用关系 (Self-Referencing Many-to-Many Relationship) 同一表中，给定记录与一个或多个其他记录相联系且一个或多个记录也能自行与该记录产生关系，则这个表存在多对多自引用关系。

一对多自引用关系 (Self-Referencing One-to-Many Relationship) 同一表中，给定记录与一个或多个其他记录相联系，则这个表存在一对多自引用关系。

一对一自引用关系 (Self-Referencing One-to-One Relationship) 同一表中，给定记录仅与另一记录相联系，则这个表存在一对一自引用关系。

自引用关系 (Self-Referencing Relationship) 一表中记录之间存在的一种关系。与两表间关系相似，自引用关系分为一对一、一对多和多对多。

集合结构 (Set Structure) 网状数据库中，一种建立和表示一个关系的透明化构造。

集合论 (Set Theory) 关系模型的两个基础数学分支之一。

SQL (Structured Query Language, 结构化查询语言) 一种标准化的语言，用于创建、维护、修改和查询关系数据库。

静态数据 (Static Data) 从来不会（或几乎不）出现变化的数据。

结构完整性 (Structural Integrity) 一套规则或指南，规定字段、表和视图定义的方式。

结构化查询语言 (Structured Query Language) 见 SQL。

子集表 (Subset Table) 一种表, 用于表示特定数据表附属主题。

表 (Table) 数据库的主要结构。它由字段和记录组成, 始终表示单个具体的主题。

表描述 (Table Description) 包括两部分: 该表所表示的主题的明确定义; 该主题对企业的重要性。

表层次完整性 (Table-Level Integrity) 数据完整性的一种, 能避免表出现重复记录, 确保表的主键值为唯一值, 不为 null 且专门识别该表的记录。

元组 (Tuple) 关系模型中等同于记录。

参与类型 (Type of Participation) 关系数据库中, 一个表参与给定关系的方式。参与类型分为强制和随意两种。

关系类型 (Type of Relationship) 给定两个表相联系的方式 (分为一对一、一对多和多对多)。

未知值 (Unknown Value) 特定字段尚未确定或定义的一个值。

URL 统一资源定位符 (Uniform Resource Locator) 的英文缩写。它表示给定资源的在线网址, 比如 www.ForMereMortals.com。

验证表 (Validation Table) 表的一种, 存储数据专用于实现数据完整性 (也称为查找表)。

验证视图 (Validation View) 视图的一种, 专用于实现数据完整性。

视图 (View) 一种虚拟表, 由其在数据库中的一个或多个基表的字段构成。

视图规范 (View Specification) 表示一个视图的所有特征, 比如名称、类型、基表等。

WAN 见广域网, (Wide Area Network)。

网页 (Web Page) 一个文件, 由一个超文本标记语言 (HTML) 文件

和可通过互联网访问的相关支持文件组成。

广域网 (Wide Area Network, WAN) 在一片宽广的地理区域中, 借助各种通信设备共享服务和资源的计算机组及其外围设备。

零长度字符串 (Zero-Length String) 两个连续单引号, 中间无空格。

参考文献

Codd, E. F. (1990) "Relational Philosopher." *DBMS*. December 1990, 34-40, 60.

——. (1990) *The Relational Model for Database Management Version 2*. Reading, MA: Addison-Wesley.

Connolly, Thomas, and Carolyn Begg. (2002) *Database System: A Practical Approach to Design, Implementation and Management, Third Edition*. Boston: Addison-Wesley.

Date, C. J. (1994) "According to Date: Many Happy Returns!" *Database Programming and Design*. September 1994, 19-22.

——. (2000) *An Introduction to Database Systems, Seventh Edition*. Boston: Addison-Wesley.

Fleming, Candace C, and Barbara von Halle. (1989) *Handbook of Relational Database Design*. Reading, MA: Addison-Wesley.

Hoffer, Jeffrey A., Mary B. Prescott, and Fred R. McFadden. (2002) *Modern Database Management, Sixth Edition*. Upper Saddle River, NJ: Prentice Hall.

Kalman, David. (1994) "Moving Forward with Relational." *DBMS*. October 1994, 62-74, 109.

Kroenke, Dr. David M. (2000) *Database Processing Fundamentals, Design and Implementation, Seventh Edition*. Upper Saddle River, NJ: Prentice Hall.

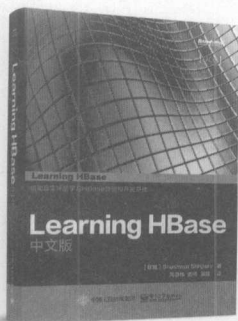
McGoveran, David. (1994) "The Relational Model Turns 25." *DBMS*. October 1994, 46-61.

Pascal, Fabian. (2000) *Practical Issues in Database Management: A Reference for the Thinking Practitioner*. Boston: Addison-Wesley.

Stephens, Ryan K., and Ronald R. Plew. (2001) *Database Design*. Indianapolis: Sams.

Teorey, Toby J. (1999) *Database Modeling & Design, Third Edition*. San Francisco: Morgan Kaufmann.

博文视点数据库最新最热图书

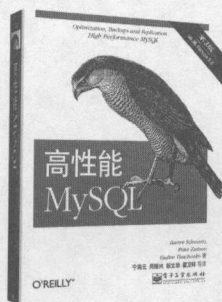


《Learning HBase 中文版》

[印度] Shashwat Shripav 著
周彦伟 姜帅 蒲聪 译
ISBN 978-7-121-27072-7
定价: 65.00 元

本书是一本介绍 HBase 知识的专业书籍, 它系统地介绍了 HBase 的基本概念, 与传统关系数据库的功能和特点的对比, 自身的配置方法以及安装方法,

同时深入介绍了 HBase 的运维管理和故障处理。本书还介绍了基于 HBase 的 Java 编程方法, 以及 HBase 作为大数据工具的一些使用案例, 这些足以帮助读者更好地理解 HBase 的架构, 更顺利地在自己的项目中使用 HBase。

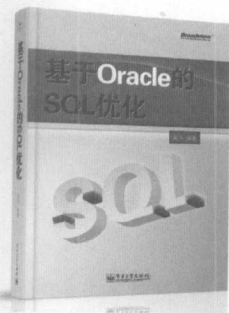


《高性能 MySQL (第3版)》

【美】Baron Schwartz, Peter Zaitsev, Vadim Tkachenko 著
宁海元 周振兴 彭立勋等 译
ISBN 978-7-121-19885-4
2013 年 5 月出版
定价: 128.00 元

MySQL 旗舰名著
惊献全面升级

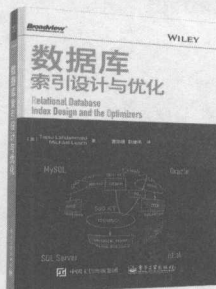
MySQL 领域的经典之作, 拥有广泛的影响力。第 3 版更新了大量的内容, 涵盖了最新 MySQL 5.5 版本的新特性, 也讲述了关于固态硬盘、高可扩展性设计和云计算环境下的数据库相关的新内容。



《基于 Oracle 的 SQL 优化》

崔华 编著
ISBN 978-7-121-21758-6
2014 年 1 月出版
定价: 128.00 元

- ◎本土 Oracle 数据库性能优化顶级大师泣血力作
- ◎集数十年实战修行与潜心钻研之大成
- ◎囊括数据库性能优化技术所有分支与脉络, 讲解通俗, 实例经典



《数据库索引设计与优化》

Relational Database Index Design and the Optimizers
【美】Tapio Lahdenmaki Michael Leach 著
曹怡倩 赵建伟 译
ISBN 978-7-121-26054-4
2015 年 6 月出版
定价: 79.00 元

盖国强、姜承尧、金官丁、叶金荣、童家旺——众国内数据库界巨腕争相作序
支付宝、网易、云和恩墨联 DB 掌门连袂推荐的造是什么书吗

编辑推荐

中国数据库界几大势力云集于这本旷世奇作, 没读过咋好意思和 DBA 同行打招呼

蚂蚁 (原支付宝) 数据库团队资深专家携成长回忆与技术历程倾情献上最优质翻译

本书旨在——通过设计适用于现代硬件的索引, 来提升关系型数据库的性能

软硬件发展让数据库性能被忽视, 但数据处理量增长更快, 全新索引优化设计才能根治随机读速缓慢

欢迎投稿:

投稿邮箱: fulm@phei.com.cn

读者信箱: market@broadview.com.cn

电话: 010-51260888

更多信息请关注:

博文视点官方网站:

<http://www.broadview.com.cn>

博文视点官方微博:

<http://t.sina.com.cn/broadviewbj>

博文视点经典好书，欢迎广大作译者朋友投稿

联系人：符隆美

联系方式：fulm@phei.com.cn

简单、易懂的数据库设计指南读物！

迈克尔·J·埃尔南德斯（Michael J. Hernandez）的畅销书《数据库设计入门》以通俗易懂的语言讲述了关系数据库设计的最简单、最高效的方法，在国际上享有盛誉。如今，他推出了新的版本，这本实用的、基于软件的设计指南变得更容易了，且该版针对上一版中提出的设计方法进行了修缮，以确保书中提出的设计方法依然适用最新的数据库和应用的设计，并且与最佳的设计行为不脱节。

本书逐步展示了设计结构健全、可靠、灵活，且适用当前网络应用的数据库的方法。从数据库思路设计到如何确定主键、设置字段说明、建立表关系、确立业务规则、建立视图等数据库设计的所有步骤都在书中一一进行了呈现和展示。通过阅读此书，你将学会确保数据完整性以及避免犯常规错误的实用方法，并学会适时打破规则。

内容包括

- 了解数据库类型、模型和设计术语
- 了解好的数据库设计能为你做什么，而糟糕的数据库设计为什么会让你的生活陷入困境
- 确立数据库要实现的目标，并将这些目标转换成真实的设计
- 分析一个现有数据库，以确定改善方法
- 确立表结构和表关系、分配主键、设置字段说明和建立视图
- 确保各层次的数据完整性
- 确定和确立业务规则

不管你采用的是何种关系数据库系统，作者都会帮助你设计一个健全且性能稳定的数据库。第一次设计数据库？补救糟糕的设计？现有数据库需要完善？

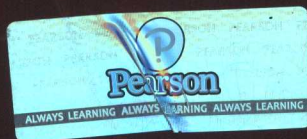
从本书开始，一一解决这些问题！

PEARSON

www.pearson.com



策划编辑：张春雨 符隆美
责任编辑：白涛
封面设计：侯士卿



上架建议：数据库

ISBN 978-7-121-26532-7



定价：99.00元